SPATIAL AND SOCIAL DIFFUSION OF INFORMATION AND INFLUENCE: MODELS AND ALGORITHMS

A Thesis
Presented to
The Academic Faculty

by

Myungcheol Doo

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computer Science

Georgia Institute of Technology
Aug 2012

SPATIAL AND SOCIAL DIFFUSION OF INFORMATION AND INFLUENCE: MODELS AND ALGORITHMS

Approved by:

Ling Liu, Committee Chair
School of Computer Science
Georgia Institute of Technology

Ling Liu, Advisor
School of Computer Science
Georgia Institute of Technology

Lakshmish Ramaswamy
Department of Computer Science
The University of Georgia

Shamkant B. Navathe
School of Computer Science
Georgia Institute of Technology

Edward Omiecinski
School of Computer Science
Georgia Institute of Technology

Calton Pu
School of Computer Science
Georgia Institute of Technology

Date Approved: May 14 2012

To my lovely wife, Misun Kang, who supported me each step of the way and

To my parents, who devoted their lives to make me successful.

ACKNOWLEDGEMENTS

This dissertation would not have been possible without my committee members, help from friends, and support from my family and wife.

First and foremost, I would like to express my deepest gratitude to my advisor, Professor Ling Liu, for her excellent guidance, caring, patience, and providing me with an excellent atmosphere for doing research. Professor Liu has been my inspiration as I hurdle all the obstacles in the completion this research work.

I also wish to thank Professor Sham Navathe, Edward Omiecinski, Calton Pu, and Lakshmish Ramaswamy for taking part in my Ph.D. defense committee. Their critical reading and penetrating remarks have made the final revisions of this dissertation.

I with to take this opportunity to acknowledge the financial subsidies I received from NSF NetSE, NSF CyberTrust, IBM SUR grant, Intel Research Council, DARPA for their support of my research work.

I would also like to thank my parents, my brother. They were always supporting me and encouraging me with their best wishes.

Finally, I would like to thank my wife, Misun Kang. She was always there cheering me up and stood by me through the good times and bad.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

With the ubiquity of broadband, wireless and mobile networking and the diversity of user-driven social networks and social channels, we are entering an information age where people and vehicles are connected at all times, and information and influence are diffused continuously through not only traditional authoritative media such as news papers, TV and radio broadcasting, but also user-driven new channels for disseminating information and diffusing influence. Social network users and mobile travelers can influence and be influenced by the social and spatial connectivity that they share through an impressive array of social and spatial channels, ranging from friendship, activity, professional or social groups to spatial, location-aware, and mobility aware events.

In this dissertation research, we argue that spatial alarms and activity-based social networks are two fundamentally new types of information and influence diffusion channels. Such new channels have the potential of enriching our professional experiences and our personal life quality in many unprecedented ways. For instance, spatial alarms enable people to share their experiences or disseminate certain points of interest by leaving location-dependent greetings, tips or graffiti and location dependent tour guide to their friends, colleagues and family members. Through social networks, people can influence their friends and colleagues by the activities they have engaged, such as reviews and blogs on certain events or products. More interestingly, the power of such spatial and social diffusion of information and influence can go far beyond our physical reach. People can utilize user-generated social and spatial channels as effective means to disseminate information and propagate influence to a much wider and possibly unknown range of audiences and recipients at any time and in any location. A fundamental challenge in embracing such new and exciting ways of information diffusion is to develop effective and scalable models and algorithms as enabling technology and building blocks. This dissertation research is dedicated towards this ultimate objective with three novel and unique contributions.

First, we develop an activity driven and self-configurable social influence model and a suite of computational algorithms to compute and rank social network nodes in terms of activity-based influence diffusion over social network topologies. By activity driven we mean that the real impact of social influence and the speed of such influence propagation should be computed based on the type, the amount and the time window of the activities performed by a social network node in addition to its social connectivity (social network topology). By self-configurable we mean that the diffusion efficiency and effectiveness are dynamically adapted based on the settings and tunings of multiple spatial and social parameters such as diffusion context, diffusion location, diffusion rate, diffusion energy (heat), diffusion coverage and diffusion incentives (e.g., reward points), to name a few. We evaluate our approach through datasets collected from Facebook, Epinions, and DBLP datasets. Our experimental results show that our activity based social influence model outperforms existing topology-based social influence model in terms of effectiveness and quality with respect to influence ranking and influence coverage computation.

Second, we further enhance our activity based social influence model along two dimensions. At first, we use a probabilistic diffusion model to capture the intrinsic properties of social influence such that nodes in a social network may have the choice of whether to participate in a social influence propagation process. We examine threshold based approach and independent probabilistic cascade based approach to determine whether a node is active or inactive in each round of influence diffusion. Secondly, we introduce incentives using multi-scale reward points, which are popularly used in many business settings. We then examine the effectiveness of reward points based incentives in stimulating the diffusion of social influences. We show that given a set of incentives, some active nodes may become more active whereas some inactive nodes may become active. Such dynamics changes the composition of the top-k influential nodes computed by activity-based social influence model. We make several interesting observations: First, popular users who are high degree nodes and have many friends are not necessarily influential in terms of spawning new activities or spreading ideas and information. Second, most influential users are more active in terms of their participation in the social activities and interactions with their friends in the social

network. Third, multi-scale reward points based incentives can be effective to both inactive nodes and active nodes.

Third, we introduce spatial alarms as the basic building blocks for location-dependent information sharing and influence diffusion. People can share and disseminate their location based experiences and points of interest to their friends and colleagues in the form of spatial alarms. Spatial alarms are triggered and delivered to the intended subscribers only when the subscribers move into the designated geographical vicinity of the spatial alarms, enabling delivering and sharing of relevant information and experience at the right location and the right time with the right subscribers. We studied how to use locality filters and subscriber filers to enhance the spatial alarm processing using traditional spatial indexing techniques. In addition, we develop a fast spatial alarm indexing structure and algorithms, called Mondrian Tree, and demonstrate that the Mondrian tree enabled spatial alarm system can significantly outperform existing spatial indexing based solutions such as R-tree, k-d tree, Quadtree.

This dissertation consists of six chapters. The first chapter introduces the research hypothesis. We describe our activity-based social influence model in Chapter 2. Chapter 3 presents the probabilistic social influence model powered with rewards incentives. We introduce spatial alarms and the basic system architecture for spatial alarm processing in Chapter 4. We describe the design of our Mondrian tree index of spatial alarms and alarm free regions in Chapter 5. In Chapter 6 we conclude the dissertation with a summary of the unique research contributions and a list of open issues closely relevant to the research problems and solution approaches presented in this dissertation.

CHAPTER I

INTRODUCTION

Internet and Web continuously transform the way how information is generated, consumed and disseminated in both enterprise organizations and our individual life. Unlike 40 years ago where the source of information was limited to classical medias such as newspapers, TV, and Radio, today a large amount of information is being generated at an astounding speed by both media companies and anyone who is connected to the Internet, thanks to the ubiquitous Internet connectivity and the affordable digital storage. Information explosion has been one of the biggest challenges to human mankind. Many of us are excited and at the same time overwhelmed by the exponential growth of information generated from increasing number of sources. Spatial and social diffusion of information and influence are no longer constrained by the physical and geographical proximity. Cyber social networks, such as Twitter [11] and Facebook [2], are examples of the most popular means of diffusing information. In conjunction with advances in wireless communication and ubiquity of mobile computing devices, such as smart phones and portable tablets, information is generated and consumed by people of all ages at all time. Data growth is faster than intelligence. We need fast and effective information retrieval systems and algorithms to find needles in the sea of information. We need efficient methods and models to disseminate the right information to the right people at the right time. We need to learn and understand how the growing availability of information and the growing connectivity of people through Internet, Web, Social Networks and mobile devices will change the way we communicate and learn, the way we influence one another, and the way we create and innovate.

This dissertation research is dedicated to the investigation of architectures, models and algorithms for information diffusion through social and spatial influence channels. We argue that information and influence are disseminated through either popular social channels, such as friends, colleagues in the physical world and connections of online social networks in the

virtual world, or relevant spatial channels, such as the places that we visit and the people whom we meet and interact at different places. Although information can be disseminated through both social channels and spatial channels, the concrete ways in which information is diffused and disseminated in the physical space can be quite different from the social and virtual space. In this dissertation, we identify and examine a number of important issues in information diffusion over the online social networks and the spatial physical world we travel from one location to another and one place to another, with the focus on scalability and effectiveness of information diffusion through social and spatial influence channels.

1.1 Information Diffusion via Social Influence

Social influence refers to the impact of a group of people on an individual member of the group by their opinions or their actions. Social influence may take many forms. For example, informational influence is an influence that is formed by accepting information from others as evidence about reality. Informational influence comes into play when people are uncertain due to some social disagreement or inherent ambiguity. Majority influence is an influence to conform to the expectations of a majority of the others that are socially connected. Generally speaking, social influences that represent positive expectations refer to those that have positive social impact on the world and the life in which we live. For instance, social influences in the form of leaderships, sales, marketing, promotion of opinions, or even peer pressure are examples of positive influence. On the other hand, social influences in the form of rumors, gossips, bullying, cussing are examples of negative influences. In the physical world in which we live, Social influence often takes place in a social context, such as family members, college friends, colleagues or community clubs. Similarly, in a cyber world, social influence takes place in a social network where people connect one another through connections in different online communities.

A social network is typically modeled as a graph of people nodes connected through friendship relationships or interactions among people. Facebook, LinkedIn [6] and Twitter are popular social networks that not only serve as a meeting point or bridging point for

many people but also an important medium for the spread of information, ideas, innovations, and influences among its members. An important property of social influence is the dynamics in terms of how influence evolves and make significant inroads into the population of the network or dies out quickly and silently. Although network diffusion processes have a long history of study in social sciences [46, 29, 58, 32, 78, 53, 24, 61], it is more challenging to understand how network diffusion and dynamics of social influence in online social networks differ from physical social networks, especially to what extent people are likely to be influenced by the opinion, the action or the decision of their friends and colleagues or to what extent the "word of mouth" effects will take hold (i.e., one is being influenced by the social networks to which it belongs).

Information is spread or propagated across a social network through connectivity between nodes, each link that connects one node to another node on the network forms a "word-of-mouth" communication channel. Directed links in social network services (SNSs) could represent anything from intimate friendships to common interests. Such directed links determine the flow of information and hence indicate a user's influence on others. As reported in [28], by analyzing 5.2 billion twitter friendships, it is discovered that on average Twitter is a network with only 5 degree of separation. This means that users in Twitter are five hops away from each other on average. In 1969, Migram [88] requested people in Nebraska to write a letter to a stock broker in Boston. None of them knew each other personally and the address of the stock broker was not given. An individual X should mail to her friend Y hoping that Y or friends of Y might know the stock broker and mail it to him. By this experiment Milgram found out that the average count was 6.2 hops from source to target. In comparison, Twitters study shows that information over the online social network can be easier and more quickly propagated from source to target than the Milgram's approach.

However, it is not well understood how to effectively use the social network as an effective means for information distribution. It is unlikely (and most of time unnecessarily) that a piece of information, an idea or innovation can be (or need to be) propagated to the entire network. We face a number of key challenges in modeling the social influence diffusion

and computing the social influence rank for each social network node. For instance, it is estimated that Twitter currently has around 500 Million registered users [12]. On one hand, we want to select a small subset of nodes that are best served as the source of information diffusion, and on the other hand, with such a large social network of millions of nodes, finding a subset of k nodes that provide the maximum influence coverage is known to be an NP-hard problem. We need to develop a good greedy algorithm with strong theoretical guarantee. In addition, it is also important to identify the set of parameters of a social influence model that can impact on the effectiveness of influence diffusion.

1.2 Information Dissemination via Spatial Diffusion Channels

Spatial channels represent another important type of information diffusion methods. We learn and collect information from the places we visit. We influence others by sharing what we learn at different places. For example, FourSquare [3] provides a mobile application such that users can "check-in" a location and recommend the location of their favorite restaurant, department store, or hotel. By sharing such information and experiences, one can find valuable crowd based ranking of restaurants, department stores for best buy items, and so forth. Location based dissemination is a unique characteristics of this application.

Location based information dissemination is a corner stone for location based advertisements and location based entertainments. We argue that spatial alarms can be seen as personalized spatial diffusion channels or reminders, installed or subscribed by mobile users. They serve as personal information and influence diffusion channels to the mobile subscribers, informing or alerting mobile users of some location dependent information or experiences upon their arrival of a specified location of interest.

Time-based alarms have been pervasively used to remind us of the arrival of a future reference time point. Many of us depend on time-based alarms everyday to better manage our daily schedule. Spatial alarms extend the concept of time-based alarms by reminding us of the arrival of a future reference location of interest. An example of a spatial alarm is Locale [5], which enables one of the pre-defined cellphone ring settings upon arrival of future reference location of interest. For example, Alice sets a spatial alarm on her school.

When she arrives at the school, Locale changes the ringer setting to the pre-defined setting such as silence.

Similar to traditional database alarms that consist of a target, an event, and an action, a spatial alarm is also comprised of these three elements. The target component specifies the future reference location of interest and the distance threshold to the alarm target. We call the region that contains the alarm target as the alarm monitoring region. The event component is the movement of the mobile client. If the user moves to one place and the distance to the alarm target is smaller than the threshold, then the event occurs. The action component of the spatial alarm defines the information or commands to be disseminated to the mobile subscribers of the spatial alarm. In the above Locale example, the event is the movement of the mobile user who is the creator or owner of this spatial alarm. The target is the school that Alice goes to. The action component is to change the ringer setting.

In order to provide location-based information dissemination, we need to develop fundamental technologies, such as spatial alarms for triggering information diffusion, smart indexing structure for storing spatial data, and a suite of algorithms for reducing load in server and reducing power consumption in client.

One of the most challenging problems in scaling spatial alarm processing is to compute alarm free regions (AFR) such that mobile objects traveling within an AFR can safely hibernate the alarm evaluation process until approaching the nearest alarm of interest. We argue that maintaining an index of both spatial alarms and empty regions (AFR) in the context of spatial alarm processing is critical for scalable processing of spatial alarms. Unfortunately, conventional spatial indexing methods, such as R-tree family, k-d tree, Quadtree, and Grid, are not well suited to index empty regions.

1.3 Our Solution Approaches

1.3.1 Activity based Social Influence Model

An intuitive approach to study the process of influence diffusion over social networks is the topological diffusion model which examines how the spread of influence is carried out through the topological relationships of friendship in a social network. For example, [57]

summarizes the state of art research in social sciences into two basic diffusion models, one based on Linear Threshold model and the other based on Independent Cascade model. Both models classify people nodes into active and inactive. Linear threshold model [22, 44, 57, 68, 75] requires each node randomly choose a weight from the interval [0,1] and set a system defined threshold such that those nodes with weights below the threshold are considered inactive. An inactive node can be switched from being inactive to being active when the total weight of its active neighbors is exceeding or equals to the threshold. In contrast, the independent cascade model [42, 43, 57, 64] uses a probabilistic approach. An inactive node is randomly become active and is given one chance to activate each of its currently inactive neighbor nodes with a system supplied probability, (which is set independently of the influence history). This process repeats until no more activation is possible. Many extensions of these two basic models exist.

Unfortunately, the topology based social influence models treat all friends equally and adopt a uniform distribution for influence diffusion from a node to all its friends. Also randomly setting edge weights in the existing diffusion models fails to reflect the importance of activities in social influence propagation: a node with higher number of activities will have higher influence on its friends and a node with low level of activities will be less active.

A popular extension of topological diffusion is to use heat diffusion principle to model social influence diffusion [62, 67, 92]. However, most of the existing social influence models, to the best of our knowledge, only relies on the spatial connectivity of people nodes and ignores two important factors of social influence among people: the amount of activities carried out by a people node and the number or volume of interactions between two people nodes. We argue that people nodes that have many more activities typically have higher level of social influence on their neighbor nodes than those who are significantly less active. In addition, two people nodes that interact more frequently will have higher influence on each other than two people nodes that have not had many interactions recently. Furthermore, frequent interactions or activities in recent time window will have higher influence on neighboring nodes than larger number of interactions or activities that were occurred way back in history.

Bearing these observations in mind, we design and develop an activity-based social influence model and an InfluenceRank algorithm to find the top k most influential people in a given social network. InfluenceRank measures how many people are influenced by the given user using activity based influence model, where heat diffusion process is simulated by assigning node with more activities higher heat and by activity based weighted heat diffusion. Like a heat flow from a place with high temperature to a place with low temperature, information also flows from a socially influential user to less influential users. There are a number of challenging issues in using heat diffusion model.

The heat diffusion approach focuses on only topology of SNS. No consideration is given to the set of additional attributes in social network, such as user profile, activities, interactions, time, etc., were not considered. For example, if a node u has two friends v and w, then in the topological heat diffusion model, u's heat is distributed evenly to v and w. In other words, if u has H_u, then v and w receive $\frac{H_u}{2}$. However, in reality, u may not have the same influence on its two friends. u may have a higher influence on v than w or vice versa. The topological approach fails to address this feature. In this dissertation research we propose to develop a self-tunable influence model that takes advantage of activities and incentives. we categorize activities into non-interactive activities (NA) and interactive activities (IA). By considering two types of activities we differentiate influence on each friend. Concretely, we extend the topology-based influence model by differentiating on activities into non-interactive activities (NA) and interactive activities (IA). By considering these two types of activities, our experiments show that we can effectively differentiate influence on each friend.

Our approach has several advantages such as it not only takes advantage of topology of the social network but also takes into account of the amount of activities when computing the social influence of a node over other neighboring nodes. Our activity-based influence ranking algorithms determine the most influential nodes in a social network in terms of maximum node coverage, maximum activity coverage and minimum ratio of duplicate nodes or duplicate activities. Our InfluenceRank algorithms select top k influential users by utilizing three activity-based influence ranking criteria: (a) independent InfluenceRank, (b) local

optimal InfluenceRank, and (c) global optimal InfluenceRank. We demonstrate through analytical and experimental evaluation that the activity-based social influence model and activity-based influence ranking metrics and algorithms are effective in differentiating the people nodes who are active though with relatively smaller number of neighbors from those people nodes that have many neighbor nodes (highly connected topologically) but inactive over a given time window. We refer readers to Chapter 2 for further detail.

1.3.2 Probabilistic Social Influence with Incentives

Although activity-based social influence model performs well, we observe that it still poorly models how the social influence is diffused in the real world. For example, among u's m friends, not all are close to u. Some are best friends, some are closed friends, and others may be acquaintances. Activity-based heat diffusion model enforces u to activating all of its m friends, which does not represent the real world correctly.

To address this problem, we extend our activity-based social influence model using a probabilistic approach where the influence of a node is only propagated to a selective subset of its neighbor nodes based on how active its neighbor nodes are probabilistically. Both interactive activities and non interactive activities are utilized to define the diffusion probability of a nodes influence over its neighbors. In addition we examine how incentives can be utilized to further stimulate the diffusion process. Concretely, with multi-scale reward points as incentives, we categorize users into five groups. The first type of users is extremely active and eager to adopt innovations earlier than others. The last type of users is inactive in adopting innovations. The other three types are defined between the first and the last type. Users may stop propagating information if a probability to activate a friend is too low. For the nodes that are inactive, we show that by adequately provisioning of incentives, inactive nodes may become active and active nodes may increase their activeness. Our experiments, conducted on Facebook datasets, Epinions datasets and DBLP datasets, show that reward points based incentives are effective.

1.3.3 Spatial Information Dissemination with Spatial Alarms

We categorize spatial alarms based on two criteria: the publish-subscribe scope of the alarms and the motion characteristics of alarm targets and alarm subscribers. According to the publish-subscribe scope, we consider three categories of spatial alarms: private, shared, and public. Private alarms are installed and used exclusively by the publisher. Shared alarms are installed by the publisher with a list of authorized subscribers and the publisher is typically one of the subscribers. Public alarms are usually installed with the purpose of sharing them with all mobile users who are entering the spatial regions of the alarms. Mobile users may subscribe to public alarms by topic categories or keywords, such as traffic information on highway 85 North in Atlanta" or Zagat survey of top-ranked local restaurants". Public alarms can be useful means of informing mobiles about hazardous road situations or heavy road congestion. The second categorization of spatial alarms is based on the motion characteristics of the alarm target and the alarm subscribers, which categorize spatial alarms into three classes: (1) moving subscriber with static target, (2) static subscriber with moving target, and (3) moving subscriber with moving target.

Processing of spatial alarms requires meeting two demanding objectives: high accuracy, which ensures no alarms are missed, and high scalability, which guarantees that the alarm processing is highly efficient and yet scales to a large number of spatial alarms and a growing collection of mobile users.

The conventional approach to spatial alarms involves periodic alarm evaluations at a high frequency. Each spatial alarm evaluation is conducted by testing whether the user is entering the spatial region of the alarm every t period of time. High frequency is essential to ensure that none of the alarms are missed. Though the periodic evaluation is simple, and easy to implement, it enforces to check all of the alarms regardless of the current location of users. This not only drains the power of mobile device due to continuous wake-ups of mobile clients and but also requires the server to perform many unnecessary alarm checks. Thus, periodic alarm evaluation can be extremely inefficient due to frequent evaluations of alarms and the high rate of irrelevant evaluations. This is especially true when the mobile user is traveling in a location that is distant from the spatial areas of all her location triggers, or

when all her spatial alarms are set on spatial regions that are far apart from one another.

In this dissertation, we describe a server-based approach to scalable processing of spatial alarms, aiming at optimizing the conventional approach of periodic spatial alarm processing by advocating mobility-awareness and safe period based alarm evaluation framework. Concretely, we formalize the concept of spatial alarms and the problem of spatial alarm processing. We introduce the concept of safe period to minimize the number of unnecessary spatial alarm evaluations, increasing the throughput and scalability of the system. We show that our safe period-based alarm evaluation techniques can significantly reduce the server load for spatial alarm processing compared to the periodic evaluation approach, while preserving the accuracy and timeliness of spatial alarms. Furthermore, we develop a suite of spatial alarm grouping techniques based on spatial locality of the alarms and motion behavior of the mobile users, aiming at optimizing the safe period computation at the server and scaling the safe-period based spatial alarm processing to a large number of spatial alarms and a growing collection of mobile users. We evaluate the scalability and accuracy of our approach using a road network simulator and show that the proposed mobility-aware safe period-based framework for spatial alarm processing offers significant performance enhancements for the alarm processing server while maintaining high accuracy of spatial alarms, especially compared to the conventional periodic alarm evaluation approach.

1.3.4 Scaling Spatial Alarms with Mondrian Tree Index

Hibernating mobile devices in spatial alarm processing is important in terms of battery power consumption. For example, if the user is far from the alarm monitoring region, all alarm evaluations are unnecessary and will not result in alarm notification. An ideal approach is to hibernate the mobile device when the user is traveling locally and only trigger it when the user is approaching the alarm monitoring region within the distance below the threshold. Thus, an important challenge in scaling spatial alarm processing is to compute alarm free regions (AFRs) such that mobile objects traveling within a rectangular region containing no spatial alarms can safely hibernate the alarm evaluation until approaching the nearest alarms of interest.

We argue that in the context of spatial alarm processing, spatial alarms and AFRs are equally important spatial data of interest and both should be treated as the first class citizen. Computing AFRs takes a significant amount of time. Thus, in this dissertation we argue that maintaining an index of both spatial alarms and empty regions (AFRs) is critical for scalable processing of spatial alarms. Unfortunately, conventional spatial indexing methods, such as R-tree family, k-d tree, and Quadtree, are not well suited to index empty regions. We present Mondrian Tree, a region-partitioning tree for indexing both spatial alarms and alarm free regions.

Mondrian tree index has two unique features. First, it utilizes the index of pre-computed empty regions to avoid on-the-fly computation of alarm free regions based on the motion behavior of mobile subscribers. Second, it incorporates a suite of locality-aware and motion-aware optimizations to further minimize the amount of client wake-ups and the number of region-crossing checks to be performed at mobile clients. We conduct an extensive experimental evaluation and show that the Mondrian tree indexing offers fast spatial alarm processing, and it significantly outperforms existing spatial indexing methods, such as R-tree, Quadtree and k-d tree, which compute alarm free regions dynamically based on the motion behavior of mobile users. Readers may refer to Chapter 4 for further detail.

1.4 Organization of This Dissertation

The rest of this dissertation is organized as follows. Chapter 2 introduces social influence using activity-based heat diffusion. Chapter 3 studies how to enhance our model by incorporating probability and incentive mechanism for maximizing social influence. Chapter 4 discusses spatial alarms and its scalable processing algorithms. Chapter 5 introduces a new indexing technology, Mondrian Tree, and a suite of algorithms for scalable processing of spatial alarms. Finally chapter 6 summarizes this dissertation work. algorithms. Chapter 5 introduces a new indexing technology, Mondrian Tree, and a suite of algorithms for scalable processing of spatial alarms. Finally chapter 6 summarizes this dissertation work.

CHAPTER II

ACTIVITY-BASED SOCIAL INFLUENCE

2.1 Introduction

Social influence refers to the impact of a group of people on an individual member of the group by their opinions or their actions. Social influence may take many forms. For example, informational influence is an influence that is formed by accepting information from others as evidence about reality. Informational influence comes into play when people are uncertain due to some social disagreement or inherent ambiguity. Majority influence is an influence to conform to the expectations of a majority of the others that are socially connected. Generally speaking, social influences that represent positive expectations refer to those that have positive social impact on the world and the life in which we live. For example, social influences in the form of leaderships, sales, marketing, promotion of opinions, or even peer pressure are examples of positive influence. On the other hand, social influences in the form of rumors, gossips, bullying, cussing are examples of negative influences.

A social network is typically modeled as a graph of people nodes connected through friendship relationships or interactions among people. Facebook, LinkedIn and Twitter are popular social networks that not only serve as a meeting point or bridging point for many but also an important medium for the spread of information, ideas, innovations, and influences among its members. An important property of social influence is the dynamics in terms of how influence evolves and which type of influence makes fast and persistent inroads into the population of the network or dies out quickly and silently. Although network diffusion processes have a long history of study in social sciences [30, 49, 54, 55, 79, 89], it is more challenging to understand how dynamics of social influence in online social networks differ from general model of network diffusion and differ from the physical social networks of friends or colleagues. In physical world, we are influenced by the friends and colleagues with whom we interact frequently and we are also influenced by the authoritative sources

such as reputable news channels like CNN, TBS. However, with online communities, it is widely recognized that people are influenced not only by those we know well but also those we met long time ago or never met before. Can we statistically compute social influence and understand quantitatively or qualitatively to what extent people are likely to be influenced by the opinion, the action or the decision of their friends, friends of friends (acquaintances or colleagues with whom we have less active and direct interactions), and strangers which we do not know in person. An in-depth understanding of such social influence and the diffusion process of such social influence will help us better address the question: to what extent the "word of mouth" effects will take hold (i.e., one is being influenced by the social networks to which it belongs) on social networks.

2.1.1 Topological Diffusion Model

An intuitive approach to study the process of influence diffusion over social networks is the topological diffusion model. It examines how the spread of influence is carried out through the topological relationships of friendship among people nodes in a social network. For example, Kempe [57] characterizes the state of art influence research in social sciences into two basic diffusion models: Linear Threshold model (LT) and Independent Cascade model (IC). Both models classify people nodes into active and inactive. LT model [22, 44, 57, 68, 75] requires each node u randomly to choose a weight $w(u, v)$ over an edge $E(u, v)$ from the interval [0,1]. $w(u, v)$ is the influence of u over v. Each node is either inactive or active. Then we set a system defined threshold θ which will be used to determine if a node u switches from being inactive to active. An inactive node u can be switched to active when the total weight $\sum w(v, u)$ is greater than or equal to θ, where v is one of active neighbors of u. This process ends if there are no more newly activated nodes. In contrast, IC model [42, 43, 57] uses a probabilistic approach. An active node u is given one chance to activate its inactive neighbor nodes with a probability $p_{u,v}$. This process repeats until no more activation is possible (reaching convergence).

Linear Threshold model and Independent Cascade model are simple basic models of influence and good for modeling a one-way spread of information. However, in each model,

a person who is active can influence her inactive friends and in contrast, an inactive node is not able to influence its active friends. This may not be realistic for certain types of social influence. For instance, in a real world, people are influenced by their friends regardless of their activeness. Assume that Alice is active and Bob is inactive. Alice finds a new product earlier than Bob and tells Bob about the product. Bob already knows about the product but does not know that it has been released. Thus, while Alice is talking to Bob and trying to activating Bob to adopt the product, Bob is also telling Alice what he knows about the product, which also displays that Bob is influencing Alice.

In addition, heat diffusion has been studied as another basic topological diffusion model. Mutual influence can readily be modeled as a heat diffusion process [62, 67, 92]. At initial time t_0, all nodes has zero heat. In a social network of n nodes, one node v_i is selected and given some amount of heat. At t_1, v_i diffuses its heat to all of its neighbors equally. At t_2, nodes with non-zero heat diffuse their heat to their neighbors. This routine repeats for a certain time period t. Then we know how many nodes are influenced from v_i by computing the number of nodes whose heat value is greater than or equal to a system defined value. By repeating this process for all n nodes, we find the top k most influential nodes.

2.1.2 Problems with Topological Diffusion Models

Most of the existing social influence models, to the best of our knowledge, define the influence diffusion solely based on the topological (spatial) connectivity of social network nodes. In this paper, we argue that social influence among people is not only determined by the social connectivity they have in a social network but also the amount of activities carried out by a node and the volume of interactions between two social network nodes. It is evident in real world that those people that have more activities in social network typically have higher level of social influence on their neighbor nodes and the friends of their friends than those who are significantly less active. In addition, two people who interact more frequently in a social network will have higher influence on each other than two people who have not had many interactions. Furthermore, frequent interactions or activities in recent time window

will have higher influence on neighboring nodes than a larger number of interactions or activities that were occurred way back in history. Bearing these observations in mind, in this paper we design and develop an activity-based social influence model and a suite of activity-based influence ranking algorithms. A node is said to have a higher social influence rank if its social influence coverage is largest in terms of the number of nodes being influenced by it. In this chapter we will introduce activity-based heat diffusion model and compare our approach with topology-based heat diffusion algorithms. We show that our activity-based influence algorithms compute the top k social network nodes with much higher influence coverage compared to the topological heat diffusion model and algorithms. We evaluate the effectiveness of our approach through analytical and experimental evaluation and demonstrate that the activity-based social influence model and activity-based influence scoring metrics and algorithms are highly effective in differentiating the social network nodes that are active even though they may have relatively smaller number of neighbors from those people that have many neighbor nodes (highly connected topologically) but are inactive.

The rest of the chapter is organized as follows. Section 2.2 gives an overview of related work, which also establishes the background for understanding and comparing the activity-based social influence model with existing topology-based social influence diffusion models. Section 2.3 describes the basic concepts and the conceptual design of our activity based social influence model, including three key social network attributes that are important for designing our activity-based social influence model and the suite of influence ranking algorithms. To make the discussion self-sufficient, in Section 2.3.2 we briefly outline the topology-based heat diffusion model of social influence. We introduce the activity based social influence model in Section 2.3.3 and describes how we extend the heat diffusion process to incorporate node activity based influence diffusion, node interaction based influence diffusion, and the hybrid influence diffusion by combining node activity with node interaction. In Section 2.4 we present independent influence ranking, locally optimized influence ranking, and globally optimized ranking and the algorithms for maximizing the spread of influence through a social network based on these influence ranking metrics. We report our experimental evaluation results in Section 2.5 and conclude in Section 2.6.

2.2 Related Work

In this section we give a brief overview of the related work on social influence from three perspectives: social influence research in social science, social influence in viral marketing, and ranking algorithms in social network services.

2.2.1 Social Influence in Social Science

The definition of influence from Oxford English dictionary is (a) the capacity to have an effect on the character, development, or behavior of someone or something, or (b) a person or thing with such a capacity or power. For example, celebrities or political leaders are considered as highly influential people. Fraser [38] interviewed Elvis Presley fans and found that they were role modeling his values and changing their own lifestyles to emulate him.

Influence still plays an important role in a social network. Coleman et al [55] showed that among physicians with higher number of academic citation, they accept a prescription of new drugs, while among doctors with lower citation the acceptance of new drug significantly depends on the fact if they interacted with more prominent peers or not. In Milgram's letter distribution experiment [88], students in Nebraska are required to send a letter to a stockbroker in Boston by passing the letters to anyone else that they believed to be socially closer to the target and on average. We note that just three friends of the stockbroker provided the final link for half of the letters that arrived successfully. Rogers [82] shows that people are affected in the adoption of individual innovations and products through peers in a social network. Gladwell [41] explained how the new ideas are transmitted by social influence. There are innovators who accept the new idea first. Innovators influence early adopters, followed by the early majority. By this time, a significant number of people adopted the new idea and the new idea becomes norm in the social network. So these early majority encourages others to conform as well. The last part of the social network to accept this new idea is called the laggards.

2.2.2 Social Influence in Viral Marketing

Viral marketing refers to the marketing techniques that utilize the word of mouth effect to produce increases in brand awareness or to achieve other marketing objectives (such as product sales, opinion spreading). In the context of social networks, viral marketing refers to marketing techniques that utilize the network effect of pre-existing social networks through self-replicating viral processes, analogous to the spread of viruses. One interesting question that is frequently asked in the context of viral marketing is: *Given a social network of people and a limited advertising budget, who should be the marketing targets such that they can have the maximum influence over the rest of the social network in terms of promoting new products to be adopted successfully.* One way to measure the maximum influence is to compute the maximum coverage of the number of influenced people.

Researchers have investigated viral marketing and social influence in order to answer the question above. Kelman [56] proposed three processes of attitude change by the social influence: *compliance*, *identification*, and *internalization*. Compliance occur when an individual accepts influence because she hopes to achieve a favorable reaction from another person or avoid punishment not because he believes in its content. Identification is to accept influence because he wants to establish a satisfying self-defining relationship to another person. Internalization is to accept because the content of the induced behavior is intrinsically rewarding. Therefore social influence can be viewed as a combined function of (a) the relative importance of the anticipated effect, and (b) the relative power of the influencing agent, and (c) the prepotency of the induced response.

Latane [63] described social impact as a phenomenon in which people affect one another in social situations. Therefore social impact consists of three rules: *social forces*, *psychosocial law*, and *multiplication of impact*. Social force is the amount of impact experienced by the target and is a multiplicative function $f = SIN$, where S is the strength of source, I is the immediacy between the source and the target, and N is the number sources present. The strength S is a measure of how much power that target perceives the source to possess. The immediacy I represents how recent the event occurred. The number of sources N is the number of sources influences the target. Hence, there will be more social impact when

individuals with high power are the sources, when the event is more recent, and when there are a greater number of sources.

Bass [20] proposed a model in which the rate of adoption is a function of the current proportion of the population who has already adopted. According to his model, the adoption rate is slow at first and speeds up at an exponential speed later.

Domingos and Richardson [33, 81] model the customer's network value using a Markov random field, which is the expected profit from sales to other customers in the network, with whom this customer may have had direct or indirect influence.

2.2.3 Ranking Algorithms in Social Networks

Due to recent arise of social network services (SNSs), such as Twitter and Facebook, researchers have proposed various social influence measurements over SNS.

One straightforward approach to influence rank is to adopt ideas from PageRank [74]. Similar to PageRank where a page has a higher rank value if more pages with high page rank values are linked to it, the rank of a social network node is measured based on the topological connectivity in the same manner as a web page is ranked. TwitterRank [91] measures the influence by taking into account both the topical similarity between users and the link structure. On the other hand, Anger [15] reveals that highly reciprocal social network structure cannot be observed with the top 10 Twitter users in Austria. This is due to the asymmetric phenomenon of social influence. The most popular super hubs are followed by many users, but the super hubs are seldomly following those users who are their followers.

PeerIndex [8] score is a relative measure of social influence. It consists of three scores: authority score, audience score, and activity score. Authority score is the measure of trust, audience score is indication of how many people are affected by the user, and activity score is the measure of how much the user is related to a topic. Note that the activity score here measures the relevance to a topic rather than the amount of actual activities of social network nodes.

Klout [5] is an influence score ranging from 1 to 100 with higher scores representing

a wider and stronger sphere of influence. Klout also measures three scores: true reach, amplification, and network impact. True reach means how many people you influence, amplification is a measure of how much you influence them, and network impact is the influence of your network. Again the Klout score depends solely on the topological structure of the social network and disregards the importance of the level of activities performed by a node and by its friends or friends of friends.

Ma et al [67] and Bao et al [19] studied social influence in terms of the heat diffusion phenomena such that the influence spreading over the social network can be modeled as heat flowing along the links from a social network node to another at some diffusion rate. After several iterations of such topological heat diffusion, those users who spread heat most are selected as the most influential nodes in the give social network. Unfortunately, the heat diffusion process utilizes only the topological structure of the given social network. As a result, social network nodes with less neighbor but more activities will receive lower ranking values.

In summary, most of the existing approaches to ranking social network nodes are primarily based on the friend relationships. Although the topological social network structure is an important indicator, many other attributes of the social network nodes, such as the number of activities performed by each node, the number of interactive activities performed by a pair of nodes, are equally important when the goal of ranking is to discover and select the most influential nodes in a given social network. In the next section, we discuss some important social network attributes and how these attributes are used to define a novel activity based social influence model − ActivityInfluence.

2.3 ActivityInfluence *Model*

In this section we introduce the basic concepts, notations, and the reference model for our activity based social influence model. We first describe the key social network attributes that are important for modeling activity-based social influence diffusion process and the basic concepts and formulation of heat diffusion based influence propagation in terms of topological structure. Then we formally introduce our ActivityInfluence model. Our

activity-based social influence model has three unique features. First, we define activity-based influence in terms of both interactive activities and non-interactive activities. Second, we model activity-based social influence as heat value at each node. Third, we use activity-based heat diffusion kernel to capture the influence propagation process in a social network.

2.3.1 Social Network Attributes

Intuitively, social network nodes that have much more activities typically have higher level of social influence on their neighbor nodes than those that are significantly less active even though they may have larger number of friends (neighbor nodes) and thus higher node degree. Thus, the number of activities that a node has performed should be an important indicator of social influence in addition to topological structure of nodes.

Furthermore, we argued that two social network nodes that interact more frequently will have higher social influence on each other than two nodes that have significantly fewer interactions recently. This leads us to introduce the number of interactive activities between two nodes as another important indicator of social influence. We call the activities performed at each node the non-interactive activities and the activities performed via interactions between a pair of nodes the interactive activities in this paper.

Third but not the least, we argue that temporal recency is another critical attribute in terms of social influence. The temporal recency effect has also been witnessed in many operational social networks today, such as Facebook [2], LinkedIn [6], and Twitter [11]. Concretely, a node u with frequent activities in recent time window will have higher social influence on those neighbor nodes with which it has had frequent interactions recently. In order to capture such temporal recency effect, we argue that the social influence of node u should not be diffused to its neighbors uniformly as done in existing topology-based diffusion models. Instead, the social influence a node u diffused to its neighbor and received from its neighbors should be computed based on the amount of its non-interactive and interactive activities occurred within a recent time window. This is especially true for those neighboring nodes of u, which have few interactive activities with u in the given time window, even though some of those nodes are highly connected (high degree nodes) or have

larger number of interactive or non-interactive activities in the past history.

As a final remark, we would like to point out the importance of asymmetric influence between a pair of social network nodes. In the context of topology based social influence model, the simplest way of constructing a social network graph is to create a vertex for each user and connect two vertices if any two users are friends. All the edges have an equal weight and thus the edge weight is insignificant. This approach is easy but not efficient because the equality of influence it promotes between a pair of nodes does not always exist in real world. The social influence between a pair of social network nodes are not symmetric, no matter whether the two nodes are mutual friends or one is a follower of another. For example, a user s is a famous singer and f is one of her fans. Then f is keen on being informed about s's activities but not vice versa. Similarly, two friends s and f may not have equal influence on one another if f is less active and s is highly active in terms of conducting non-interactive and interactive activities.

In the design of our ActivityInfluence model, we advocate the need of utilizing multiple social network attributes, such as timestamps, the number of individual user activities, the number of interactions between users, topological and activity based similarity of user profiles in social influence modeling and analysis in order to increase the accuracy and the effectiveness of social influence. In the rest of this chapter, we will focus on the social network attributes that are important to capture the quantity and quality of activities to define our ActivityInfluence model and its influence diffusion process, such as number of friends, number of activities (interactive and non-interactive) and timestamp of activities. We refer readers to our ongoing study on how the user profile similarity plays a role in social influence modeling and computation in a separate project [93].

Number of Friends

Measuring the social influence based on the number of friends is the easiest and the most common way to date. We refer to this type of social influence as popularity based or friendship based social influence to differentiate it from the general concept of social influence, which should capture both topological structure and activity based semantics of social network nodes. Indeed, highly influential people often have more followers than less

or non-influential people. Thus node degree (number of friends) is an important measure for computing social influence of nodes in a social network. However, the number of friends should not be used as the best and the only indicator of the level of influence. Some people report that they cannot reject friend requests from their clients [86] or they accept friend requests from even vaguely recognized people [23] to increase their popularity. Therefore, the influence diffusion model, which is based solely on the number of friends and the uniform distribution of influence among all friends of a node, will fail to capture how social influence is actually diffused in real life. As soon as we start to differentiate the influence diffused to and received from different friends of a node, activity-based attributes will take a part in such differentiation. Furthermore, the level of social influence differs from context to context. For example, a country singer may have many followers but her social influence on a deep science and engineering subject will be less significant compared to her influence on fashion and restaurants. Thus, we need to identify and utilize additional social network attributes to gauge social influence more correctly and accurately.

Activities

In reality, friendship is not the only contextual feature that most social network services provide. Users can post photos, videos, and reviews on any subject. For friends' posting, users can vote for 'like' or write a comment. We categorize user activities into two groups: *interactive* activities (such as comments on friends postings) and *non-interactive* activities (such as reviews or postings by a node). Interactive activities are user activities that involves another social network node than self. Otherwise, activities are considered as non-interactive. For example, Alice posts a photo on her profile page. This activity is non-interactive because it involves only herself and no one else. If another social network node Bob leaves a comment on Alice's photo. Then this activity is an interactive activity between Bob and Alice because it involves two nodes in the social network.

Figure 1 shows an example of social network of 10 users, $v_1, v_2, \ldots, v_{10}$. Each user is represented as a vertex. Each node may have performed a number of non-interactive activities, such as posting of reviews, photos or videos. The positive integer with underline attached to each node represents the number of non-interactive activities performed by the node. If two

users are friends, then there is an edge between two vertices. The positive integer attached to each edge connecting two nodes denotes the number of interactive activities performed between them. For example, v_1 has 70 postings, v_2 has 40 postings, and v_3 has 3 postings. There are 80 interactive activities between v_1 and v_2. However, there are only 5 activities between v_1 and v_3. This example also shows that the influence of v_1 on its neighbor nodes v_2 and v_3 likely follows a non-uniform distribution.

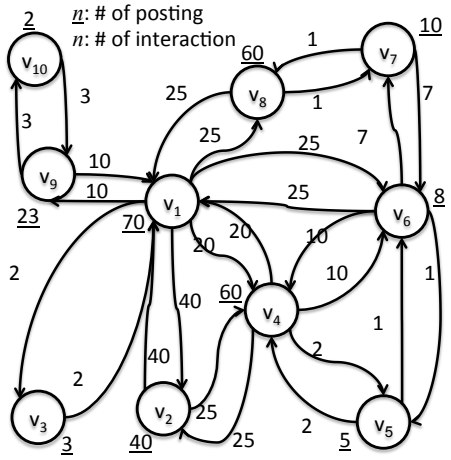


Figure 1: Example Social Network

Timestamp

We can further augment non-interactive and interactive activities with timestamp to differentiate recent activities from older historical activities. Consider two car reviews from users u and v. u left a review on a car in 2000 and v left a review on the newer version of car in 2011. People interested in the car of 2011 consider u's review more useful, and therefore more influential. Another scenarios is that v left 100 reviews in 2000 but did only two reviews in 2011. However, u posted 50 reviews in 2011. Although v has more reviews but his reviews are outdated and might be useless now. Therefore we argue that recent activities should be weighted more.

2.3.2 Heat Diffusion Kernel

Heat diffusion is a physical phenomenon such that heat always flows from an object with high temperature to an object with low temperature. In physics, the heat capacity is a

thermodynamic property; i.e., the heat capacity is a measure of how much an intensive thermodynamic variable (temperature) changes when a small amount of energy is added or subtracted from the sample. Thermal conductivity is a transport property; the thermal conductivity, α, is the linear transport coefficient that relates a temperature gradient to a heat flux. In most cases of practical interest, the α is a tensor and is diagonal. The ratio of the thermal conductivity and heat capacity per unit volume, C, is the thermal diffusivity, $D = \frac{\alpha}{C}$. We can get a rough idea of the time t it takes for heat to diffuse some distance L from dimensional analysis and t is increasing with increase of L and decrease of D. Thus, the time scales of heat diffusion in practical situations varies enormously. In scientific studies, the relevant time scales of heat diffusion span an amazing 27 orders of magnitude. For example, some physicists study heat diffusion in thin films on 10 picosecond time scales and planetary scientists are concerned with the diffusion of heat on the time scale of billions of years.

In a large social network graph, heat diffusion kernel can be used to model to social influence diffusion process in the social network graph. Let $G = (V, E)$ denote a social network graph where $V = \{v_1, v_2, ..., v_n\}$ is the set of vertices representing users and $E = \{(u, v)|u, v \in V\}$ is a set of edges representing friend relationship between users. Let α denote the thermal conductivity (the heat diffusion coefficient) on G. The heat on vertex v_i at time t is represented as a function $H_i(t)$ and heat flows from high temperature node to low temperature node following the edges between vertices. On a directed graph for the duration Δt, v_i diffuses its heat, denoted by $DH_i(\Delta t)$, through its outgoing edges, and receives heat, denoted by $RH_i\ (\Delta t)$, through its incoming edges. The heat at vertex $v_i \in V$ between t and $t + \Delta t$ is defined by the sum of the differences between the heat that it receives from, and the heat it diffuses to, all its neighbors, and is formulated as follows:

$$H_i(t + \Delta t) - H_i(t) = RH_i(\Delta t) - DH_i(\Delta t) \tag{1}$$

This formula shows that a number of factors impacts on $H_i(t + \Delta t) - H_i(t)$, including the heat conductivity α, the heat at vertex v_i, the duration of heat diffusion process Δt, and the number of friends of v_i, denoted by d_i.

Each heat distribution medium has different heat conductivity α, which is a real number between 0 and 1. If α is approaching zero, then the medium barely transfer any heat. If α is approaching the upper bound of 1, it transfers heat without loss and the speed of heat diffusion is faster than when α is close to the lower bound of 0. Each social network has different speed of information diffusion and so does each type of products or opinions. In this paper we set α to be 1 to allow us to focus more on the methods to define heat at a node and heat diffusion on weighted edges by taking the assumption that the more heat a social network node v_i has, the more heat v_i can diffuse.

The amount of heat transfer depends on the topology of graph, time of period, and the amount of heat at source node. If v_i is directly reachable from v_j through an edge in $E(v_jv_i)$, then the more heat v_j has, the more heat v_i receives, and the longer the heat diffusion process takes, the more heat v_i receives from its neighbors and v_j diffuses to its neighbors. Lastly, the number of friends of v_i, denoted by d_i, is the out-degree of vertex v_i, another important factor for influence diffusion. In the topology based heat diffusion model where heat at a vertex is uniformly distributed to all its neighbors, each neighbor node of v_i receives $\frac{1}{d_j}$ of heat that v_i diffuses.

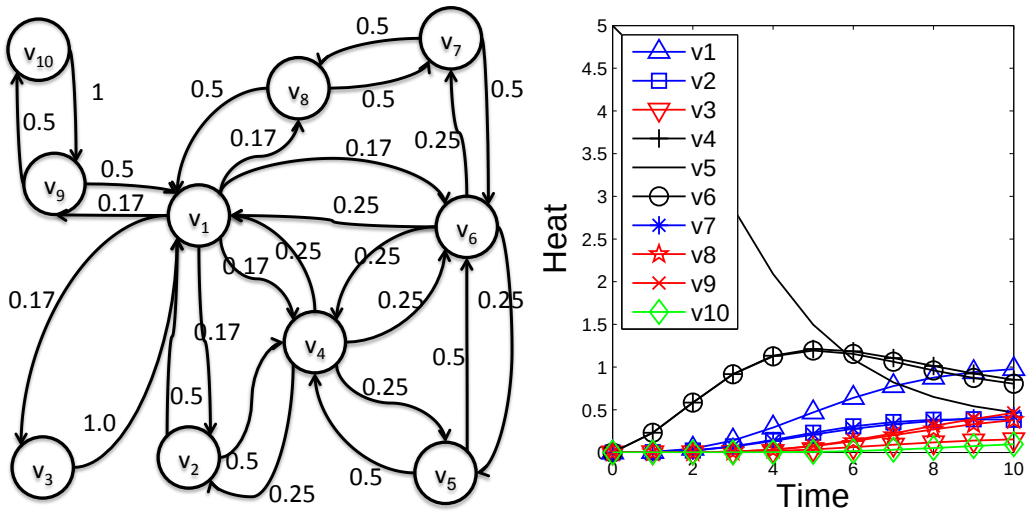
Figure 2(a) shows an example extracted from the sample social network given in Figure 1. Weight on edges are computed based on the uniform distribution, namely $\frac{1}{d_i}$ of the heat of vertex v_i will be diffused to each of its neighbors. For example, v_5 has two friends, v_4 and v_6. Hence, the weight of 0.5 is assigned to $E(v_5, v_4)$ and $E(v_5, v_6)$ respectively and one half of v_5's heat is transmitted to v_4 and the other half is transferred to v_6. v_{10} has only one friend, v_9. Therefore, the weight on $E(v_{10}, v_9)$ is 1 and all of v_{10}'s heat is diffused to v_9.

In summary, $DH_i(\Delta t)$ is proportional to α, Δt, and $H_i(t)$. $RH_i(\Delta t)$ is proportional to α, Δt, and sum of $\frac{H_j(t)}{d_j}$. we can formulate $DH_i(\Delta t)$ and $RH_i(\Delta t)$ as follows:

$$DH_i(\Delta t) = \alpha \Delta t H_i(t) \tag{2}$$

$$RH_i(\Delta t) = \alpha \Delta t \sum_{j:(v_j,v_i)\in V} \frac{H_j(t)}{d_j}. \tag{3}$$

Let $H(t)$ denote a matrix representation of heat on the set of vertices of $G = (V, E)$ at



(a) Constructed Graph (b) Heat

Figure 2: Heat Diffusion (Topology)

time t. We can plug Eq. (2) and (3) into Eq. (1) then we have the following form:

$$H_i(t+\Delta t) - H_i(t) = \alpha \Delta t \Big(\sum_{j:(v_j,v_i)\in V} \frac{H_j(t)}{d_j} - H_i(t) \Big). \tag{4}$$

Consider the example in Figure 2(a), we have:

$$\begin{aligned}
H_1(t+\Delta t) - H_1(t) &= \alpha \Delta t \Big(-H_1(t) + 0.5H_2(t) + H_3(t) + 0.25H_4(t) + 0.25H_6(t) + 0.5H_8(t) + 0.5H_9(t) \Big) \\
H_2(t+\Delta t) - H_2(t) &= \alpha \Delta t \Big(0.17H_1(t) - H_2(t) + 0.25H_4(t) \Big) \\
H_3(t+\Delta t) - H_3(t) &= \alpha \Delta t \Big(0.17H_1(t) - H_3(t) \Big) \\
&\vdots \\
H_{10}(t+\Delta t) - H_{10}(t) &= \alpha \Delta t \Big(0.5H_9(t) - H_{10}(t) \Big).
\end{aligned}$$

The above heat difference can be represented in a matrix form as follows:

$$
\begin{aligned}
H(t+\Delta t) - H(t) &= \alpha \Delta t \begin{pmatrix}
-1 & 0.5 & 1.0 & 0.25 & 0 & 0.25 & 0 & 0.5 & 0.5 & 0 \\
0.17 & -1 & 0 & 0.25 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.17 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0.17 & 0.5 & 0 & -1 & 0.5 & 0.25 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0.25 & -1 & 0.25 & 0 & 0 & 0 & 0 \\
0.17 & 0 & 0 & 0.25 & 0.5 & -1 & 0.5 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0.25 & -1 & 0.5 & 0 & 0 \\
0.17 & 0 & 0 & 0 & 0 & 0 & 0.5 & -1 & 0 & 0 \\
0.17 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.5 & -1
\end{pmatrix} H(t) \\
&= \alpha \Delta t K H(t)
\end{aligned}
\tag{5}
$$

where K is a $n \times n$ matrix whose element $K(i,j)$ is defined as in Eq. (6) and $H(t)$ is a column vector of size n defined in Eq. (7). In this example $n = 10$.

$$
K(i,j) = \begin{cases}
\frac{1}{d_j} & (v_j, v_i) \in E \\
-1 & i = j \ and \ d_i > 0 \\
0 & \text{otherwise}
\end{cases}
\tag{6}
$$

$$
H(t) = \begin{Bmatrix}
H_1(t) \\
H_2(t) \\
H_3(t) \\
\cdots \\
H_{10}(t)
\end{Bmatrix}
\tag{7}
$$

We can transform Eq. (5) as follows:

$$\begin{aligned} H(t+\Delta t) - H(t) &= \alpha \Delta t K H(t) \\ \Leftrightarrow \frac{H(t+\Delta t) - H(t)}{\Delta t} &= \alpha t K H(t) \\ \Leftrightarrow \lim_{\Delta t \to 0} \frac{H(t+\Delta t) - H(t)}{\Delta t} &= \alpha t K H(t) \\ \Leftrightarrow \frac{d}{dt} H(t) &= \alpha K H(t) \end{aligned} \tag{8}$$

.

Eq. (8) is a linear homogenous differential equation and can be solved as follows:

$$H(t) = e^{\alpha t K} H(0) \tag{9}$$

where K denotes the heat diffusion kernel of G and $H(0)$ denotes the initial heat distribution column vector at time zero on G. Eq. (9) defines the vertex's thermal capacity at time t by an exponential function $H(t)$ with independent variable t for the initial heat source $H(0)$. The matrix $e^{\alpha t K}$ is called the propagating heat diffusion kernel and can be represented as a Taylor series:

$$\begin{aligned} H(t) =& e^{\alpha t K} H(0) \\ =& (I + \alpha t K + \frac{\alpha^2 t^2}{2!} K^2 + \frac{\alpha^3 t^3}{3!} K^3 + \dots) H(0) \\ =& I + \sum_{n=1}^{\infty} \frac{\alpha^n t^n}{n!} K^n H(0) \end{aligned} \tag{10}$$

where $n!$ denotes the factorial of n and 0! is defined to be 1. $K^{(n)}$ denotes the nth derivative of K evaluated at the point t and the zeroth derivative of K is defined to be K itself.

Figure 2(b) shows the result of heat diffusion using Eq. (6) and (9). v_5 is selected as a heat source. x-axis is the time line and y-axis is the amount of heat at each node v_i. v_4 and v_6 are nodes that are 1-hop away from v_5. They evenly receive heat from v_5. Thus their heat graphs are the same. 2-hop away nodes, v_1, v_2, v_7, have less amount of heat than 1-hop away nodes but larger than 3 or more -hop away nodes. Although v_1 is 2-hop away node, it is receiving from three nodes. Thus its heat is higher than other two 2-hop away nodes.

We have introduced the basics of heat diffusion formulation and have discussed the problem inherent in the topology-based diffusion process, where a user v_i diffuses its influence evenly to all of its neighbors. We argue that the influence of a social network node v_i should be determined by taking into account of both the number of interactive activities that v_i has done with its neighbors and the number of non-interactive activities that v_i has performed.

2.3.3 Activity-based Heat Diffusion

We design the activity based social influence model by extending the basic heat diffusion kernel. By refining the heat diffusion kernel to incorporate both interactive activities between a pair of nodes and non-interactive activities performed independently at each node, we can define

2.3.3.1 Design Guidelines

Let IA_{ij} denote the number of interactive activities from node v_i to its neighbor node v_j and NIA_i denote the number of non-interactive activities at node v_i. There are several ways one can extend the heat diffusion kernel to incorporate both interactive and non-interactive activities. In the first prototype of ActivityInfluence system, we choose to extend the basic heat diffusion kernel in two steps.

First, we argue that non-interactive activities at node v_i may play a role as heat source. The more non-interactive activities a node v_i has performed, the more heat is added to v_i and consequently v_i is losing its heat at a slower pace in the diffusion process. Concretely, let NA_i denote the number of non-interactive activities at node v_i. We define $MAX(NA)$ as the largest number of non-interactive activities by node in V. Thus, the diffused influence at vertex v_i, denoted by $DH_i(t)$, is augmented by the number of non-interactive activities at v_i normalized by $MAX(NA)$, denoted by $\frac{NA_i}{MAX(NA)}$. In order to express that v_i loses its heat slower when it has larger number of NA_i, we set $DH_i(t)$ is proportional to $1 - \frac{NA_i}{MAX(NA)}$.

Second, we argue that the amount of influence received by node v_i from one of its neighbors, say v_j, should be proportional to the number of interactive activities that v_j

has performed with v_i normalized by the total number of interactive activities that v_j has performed with all of its neighbors, namely $\frac{IA_{ji}}{\sum_{k:(v_j,v_k)\in E} IA_{jk}}$. Thus $RH_i(t)$ is proportional to $\frac{IA_{ji}}{\sum_{k:(v_j,v_k)\in E} IA_{jk}}$.

Consider the example in Figure 1, v_2 has 80 interactions with v_1 and 50 with v_4. Instead of following topology based heat diffusion where v_1 and v_4 receive equal amount of heat from v_2, namely $\frac{H_2(t)}{2}$, in the ActivityInfluence model, we define that v_1 receives $\frac{80}{80+50}$ of $F_2(t)$ and v_4 receives $\frac{50}{80+50}$ of $H_2(t)$. Similarly, we can consider non-interactive activities. NA_1 is 70, NA_{10} is 2, and $MAX(NA)$ is 80. Then we can normalize such that NA_1 is normalized as $\frac{70}{80}$ and NA_{10} is normalized as $\frac{2}{80}$. Then the amount of heat v_1 loses is $(1-\frac{70}{80})H_1(t)$ and v_{10} loses $(1-\frac{2}{80})H_{10}(t)$ amount of heat Thus v_{10} loses its heat faster than v_1.

2.3.3.2 Activity based Diffusion Kernel

In this section we formally define the activity based diffusion kernel by taking into account of both interactive and non-interactive activities.

Considering the above example we can formulate as follows:

$$RH_i(\Delta t) = \alpha \Delta t \sum_{j:(v_j,v_i)\in E} H_j(t)\left(\frac{IA_{ji}}{\sum_{k:(v_j,v_k)\in E} IA_{jk}}\right) \tag{11}$$

$$DH_i(\Delta t) = \alpha \Delta t H_i(t)\left(1-\beta\frac{NA_i}{MAX(NA)}\right) \tag{12}$$

where β is a real number between 0 to 1 and served as the weight for non-interactive activities. If β is set to 0 then non-interactive activities are ignored. If β is set to 1, then $DH_i(\Delta t)$ may become zero when $MAX(NIA) = NA_i$.

By plugging Eq. (11) and (12) into the heat difference during Δt, defined in Eq. (1), we have the activity-based heat diffusion formula as follows:

$$\begin{aligned} & H_i(t+\Delta t) - H_i(t) \\ & = RH_i(\Delta t) - DH_i(\Delta t) \\ & = \alpha \Delta t \sum_{j:(v_j,v_i)\in E} H_j(t)\left(\frac{IA_{ji}}{\sum_{m:(v_j,v_m)\in E} IA_{jm}}\right) + \alpha \Delta t H_i(t)\left(1-\beta\frac{NA_i}{MAX(NA)}\right) \end{aligned} \tag{13}$$

Eq. (13) is transformed into matrix form as follows:

$$\alpha \Delta t K H(t) \tag{14}$$

, where K is a $n \times n$ matrix as defined as follows:

$$K(i,j) = \begin{cases} \frac{IA_{ji}}{\sum_{m:(v_j,v_m)\in E} IA_{jm}} & (v_j, v_i) \in E \\ 1 - \beta \frac{NA_i}{MAX(NA)} & i = j \ and \ d_i > 0 \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

We also define node activity-based diffusion kernel and node activity-based diffusion kernel. If there are only node activity information and no node interaction, then $RH_i(\Delta t)$ is the same as one in the topological heat diffusion. In concrete, we use $DH_i(\Delta t)$ in Eq. (12) and $RH_i(\Delta t)$ in Eq. (3):

$$DH_i(\Delta t) = \alpha \Delta t H_i(t)(1 - \beta \frac{NA_i}{MAX(NA)})$$
$$RH_i(\Delta t) = \alpha \Delta t \sum_{j:(v_j,v_i)\in V} \frac{H_j(t)}{d_j}.$$

. Then we modify K as follows:

$$K(i,j) = \begin{cases} \frac{1}{d_j} & (v_j, v_i) \in E \\ 1 - \beta \frac{NA_i}{MAX(NA)} & i = j \ and \ d_i > 0 \\ 0 & \text{otherwise} \end{cases} \tag{16}$$

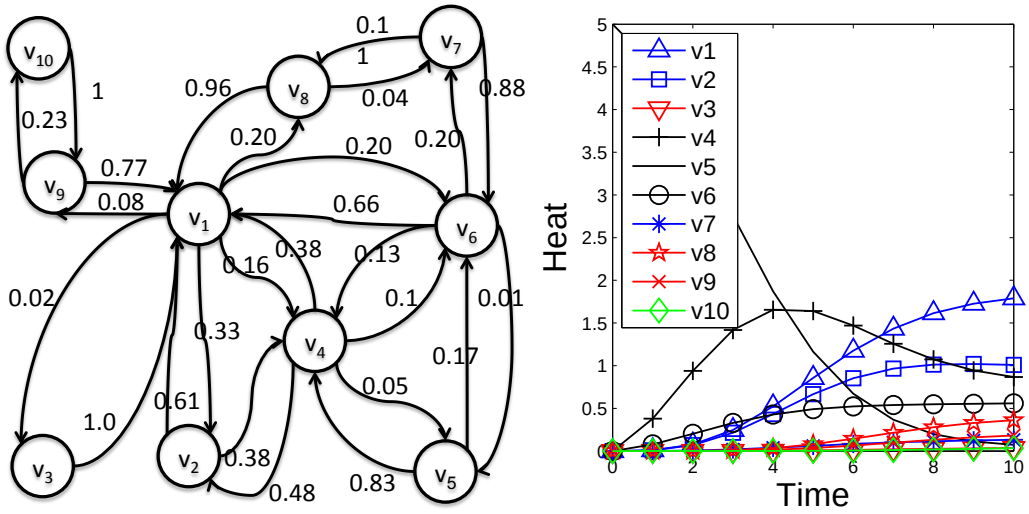
On the other hand, If there are only node interaction information and no node activity information, then $DH_i(\Delta t)$ is the same as one in the topological heat diffusion. In concrete, we use $DH_i(\Delta t)$ in Eq. (2) and $RH_i(\Delta t)$ in Eq. (11):

$$DH_i(\Delta t) = \alpha \Delta t H_i(t)$$
$$RH_i(\Delta t) = \alpha \Delta t \sum_{j:(v_j,v_i)\in E} H_j(t)(\frac{IA_{ji}}{\sum_{m:(v_j,v_m)\in E} IA_{jm}})$$

and K is also defined as follows:

$$K(i,j) = \begin{cases} \frac{IA_{ji}}{\sum_{m:(v_j,v_m)\in E} IA_{jm}} & (v_j, v_i) \in E \\ 1 & i = j \ and \ d_i > 0 \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

Figure 3(a) shows K as weights on edges computed using the above formula. Weights on Figure 3(a) is different from those in Figure 2(a). Figure 3(b) shows activity based heat diffusion result. From the heat source v_5, v_4 receives the largest amount of heat (83%) and its heat increases faster followed by v_6, which receives 17% of v_5's heat. Note that in Figure 2(a), v_4 and v_6 receive the same amount of heat from v_5, but now the ratio is changed. Also note that v_1 are v_2 have higher heat than v_6. This is because v_4 has higher heat than v_6 and v_4's heat is diffused more to v_1 and v_2 than v_6. Also v_1 is receiving heat from v_6. Therefore v_1 and v_2 have higher heat than v_6.



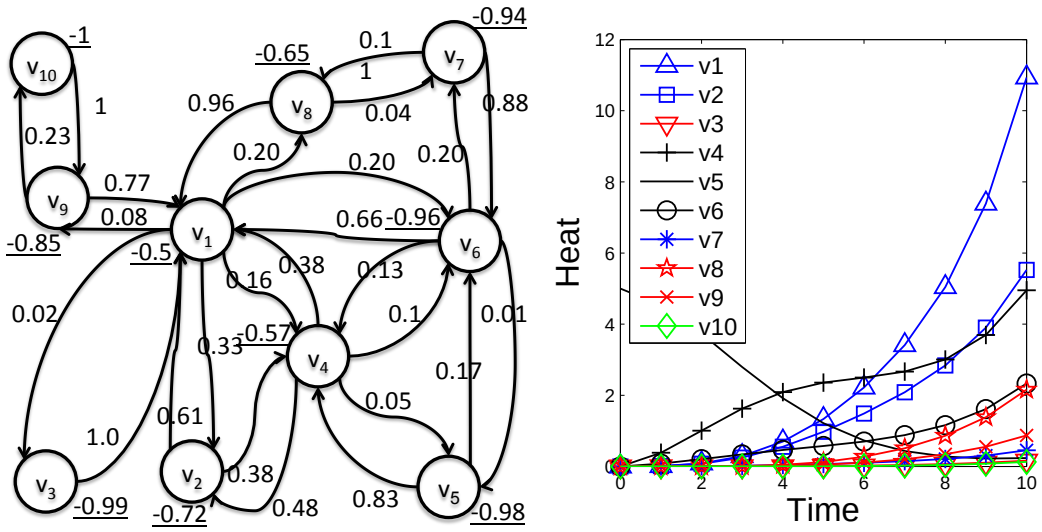
(a) Activity-DiffusionConstructed Graph

(b) Heat Diffusion over time

Figure 3: Heat Diffusion at different nodes(Interaction)

Figure 4 shows the result of activity based heat diffusion process with information on both the number of interactive activities and the number of non-interactive activities. We set β to 0.5 in order to fully consider non-interactive activities. Each node receives the same heat as in Figure 3(a) but they lose heat differently. The rate of heat loss in Figure 4(b) is slower than the one in Figure 3(b). Note that some users such as v_1 and v_2 are actively posting once they are influenced by v_4.

In summary, the topology-based heat diffusion model allows each node to lose its heat proportionally to time t, which means that each vertex loses its heat at the same rate. In our ActivityInfluence model, we utilize the number of non-interactive activities to



(a) Constructed Graph (b) Heat

Figure 4: Heat Diffusion (Non-interactive Activities)

define how fast a node loses its heat and capitalize on the number of interactive activities to define how much heat a node v_i should receive from its neighbors.

2.4 InfluenceRank, InfluenceCoverage *Algorithms*

We have described the activity-based heat diffusion model, which can be used to compute the social influence in fixed rounds of iteration. In this section we design the InfluenceCoverage, which can measure the coverage of nodes that are influenced by a given user using the heat diffusion process. We can define the InfluenceCoverage of a node v_i by the coverage of the nodes that are influenced by v_i. Given InfluenceCoverage, we assign InfluenceRank by descending order of InfluenceCoverage. This can be directly applied to viral marketing scenarios where a limited budget is allocated to some new products. Therefore marketing candidates should be carefully selected so that the marketing efficiency can be maximized.

Determining Influence Threshold.

Given heat at v_i, v_i diffuses its heat to its neighbors. At some time t, we check the heat value at every vertex. Each vertex has an acceptance threshold θ. If the heat is greater than or equal to θ we consider the user represented by influenced by the initial heat source,

v_i.

Top K Influential Nodes by InfluenceRank.

Given a social network G of size N, the top K influence rank based node selection algorithm performs the following four tasks in sequence: (a) activity based social influence computation based on activity based heat diffusion kernel, (b) InfluenceCoverage computation, (c) sorting social network nodes by InfluenceCoverage, and assigning InfluenceRank based on InfluenceCoverage, and (d) returning top-k nodes by their InfluenceRank.

It is well known that selecting top-k influential people from the network of N nodes ($k < N$) to maximize the spread of their influence is NP-hard [57]. Given that our social influence maximization function has both monotonic and diminishing return properties, thus we can present greedy algorithms for selecting top-k people in a social network graph with a strong theoretical guarantee that the greedy algorithm will approximate the theoretical optimal solution [57].

There are several ways to compute the top k most influential nodes based on InfluenceRank. For example, if we simply rank all N nodes based on their InfluenceCoverage, then the top k nodes are selected as they have the top k largest influence coverage. However, this approach may not give the highest overall node coverage by the top k nodes when there is a large overlap in terms of node coverage among the top k nodes. In this section we describe three influence ranking criteria: (a) independent InfluenceCoverage, (b) local optimal InfluenceCoverage and (c) global optimal InfluenceCoverage.

Top K by Independent InfluenceCoverage

The simplest approach to selecting top-k influential nodes in a social network is to compute their InfluenceCoverage and sort the nodes by their individual InfluenceCoverage. Algorithm 1 shows the pseudocode of the Independent InfluenceCoverage algorithm. For each v_i in G, we assign the same heat h_0 and compute InfluenceCoverage in terms of the number of nodes over which v_i has influenced.

Algorithm 1 is easy to implement but it may select top k nodes that have very high level of overlapping in terms of node coverage. Given $k = 2$, we denote a set of influenced users by v_1 as IS_1. For example, given $IC_1 = \{v_{11}, v_{12}, v_{13}\}$, $IC_2 = \{v_{11}, v_{12}, v_{13}, v_{14}\}$, and

Algorithm 1: Independent InfluenceCoverage

```
1  foreach v_i ∈ G do
2      H(0) ← 0;                                   /* initialize heat */
3      H_i(0) ← h_0;                               /* assign heat h_0 to v_i */
4      IC_i ← ∅;                                   /* initialize IC_i */
5      HeatDiffusion(t, H(0));
6      foreach v_j ∈ G do
7          if H_j(t) ≥ θ_j then
8              Add v_j into set IC_i;
9          end
10     end
11 end
12 Sort {IC_i | v_i ∈ G} by set size;
13 return top-k v_i ordered by |IC_i|
```

$IC_3 = \{v_{20}, v_{21}\}$, Algorithm 1 returns v_2 and v_1 as top 2 influential users because $|IC_2| > |IC_1| > |IC_3|$. v_2 and v_1 influence four users, which is $IC_1 \cup IC_2 = \{v_{11}, v_{12}, v_{13}, v_{14}\}$. On the other hand, v_2 and v_3 influence 6 users, which is $IC_2 \cup IC_3 = \{v_{11}, v_{12}, v_{13}, v_{14}, v_{20}, v_{21}\}$. Simply returning first k largest IC_i does not guarantee the true top-k influential people. Therefore we present the second criteria in selecting top-k people, which is minimizing the local overlap.

Top K by Locally Optimal InfluenceCoverage

Instead of selecting top k nodes with the largest individual InfluenceCoverage, the local optimal InfluenceCoverage algorithm adds node v_i to its top k list if it satisfies two conditions: (i) v_i has the high InfluenceCoverage and (ii) v_i has the minimal intersection with previously selected set of nodes as described in Algorithm 2. This algorithm extends the independent influence ranking Algorithm 1 by adding another round of computation to find the set of nodes in the social network, which minimizes local overlap, upon completion of the heat diffusion process. Each node found is added to the top-k list of influential nodes until every node in the social network is examined.

Top k By Globally Optimal InfluenceCoverage

Algorithm 1 and 2 computes InfluenceCoverage while assuming that there is only one heat source. To increase the overall influence coverage by the top k selected nodes, we need an algorithm that uses multiple heat sources to conduct the social influence computation

Algorithm 2: Minimizing Local Overlap

```
1  Line 1 to 12 in Algorithm 1;                          /* heat diffusion */
2  IC ← ∅;                          /* universal set of influenced people */
3  m ← 0;
4  V_local ← 0;
5  while m < k do
6      foreach v_i ∈ (G − V_local) do
7          Find IC_i that has max(IC_i − IC);
8      end
9      V_local ← V_local ∪ v_i;
10     m ← m + 1;
11     IC ← IC ∪ IC_i;
12 end
13 return marked IC_i
```

and influence ranking.

Algorithm 3 gives a sketch of the top k selection by global optimal influence ranking. It uses the Hill Climbing approach. In order to avoid local maximum, it performs Algorithm 1 and gets the sorted vertices by individual InfluenceCoverage. We then select v_i whose InfluenceCoverage, denoted by IC_i is the largest and initialize V_{global} by adding v_i into V_{global}. At each step in the outer loop, we give heat to vertices in V_{global}. By adding one more heat source at a time, we simulate the scenario that there are multiple heat sources. After the second inner loop, we find v_i, which has the minimal overlap with previous selected nodes in V_{global}.

In summary, the independent InfluenceCoverage algorithm selects top k nodes such that each has high influence coverage independently but the subset of the top k nodes together fails to maximize the overall social influence of the k nods due to the high degree of overlap in their InfluenceCoverage. The local optimal InfluenceCoverage algorithm provides a local greedy method to reduce the degree of overlapping among locally connected nodes in terms of InfluenceCoverage of the top k nodes. The global optimal InfluenceCoverage algorithm offers a global greedy approach, which maximizes the overall influence coverage of its selected top k nodes. Our experimental evaluation in the next section shows that the global optimal InfluenceCoverage algorithm prevails over both local greedy and independent top k algorithms for all three real datasets we used

Algorithm 3: Minimizing Global Overlap

```
1  Line 1 to 12 in Algorithm 1;                    /* single-source heat diffusion */
2  IC ← ∅; m ← 0;
3  Find v_i that has max(IS_i);
4  V_global ← v_i;
5  while m < k do
6      H(0) ← 0;
7      foreach v_i ∈ V_global do
8          H_i(0) ← h_0;                                /* multiple heat sources */
9      end
10     foreach v_i ∈ (G − V_global) do
11         H_i(0) ← h_0;
12         HeatDiffusion(t, H(0));                /* multi-source heat diffusion */
13     end
14     Find v_i that has max(IC_i − IC);
15     V_multi ← V_global ∪ v_i;
16     m ← m + 1;
17 end
18 return V_global
```

for experimental evaluation.

2.5 Experiments

We performed experimental evaluations in order to show the performance and effectiveness of the activity-based social influence against topology-based heat diffusion. First of all, we explain how we collect datasets, what properties these datasets have, what the effects of parameters, and the performance of our approaches. Our results show that activity-based approach has larger coverage so that with limited marketing budges marketing companies maximize the advertising.

2.5.1 Datasets

We used datasets from three sources, DBLP[10], Epinions[1], and and Facebook[2], to evaluate the effectiveness of our activity based social influence model and influence ranking algorithms. DBLP dataset provides bibliographic information on major computer science journals and proceedings. We parse DBLP data and extracted 5,000 authors and their co-authorship information. For example, if authors u and v wrote x number of papers together, then we create two edges $e_1(u, v)$ and $e_2(v, u)$ and set x as the number of interactive

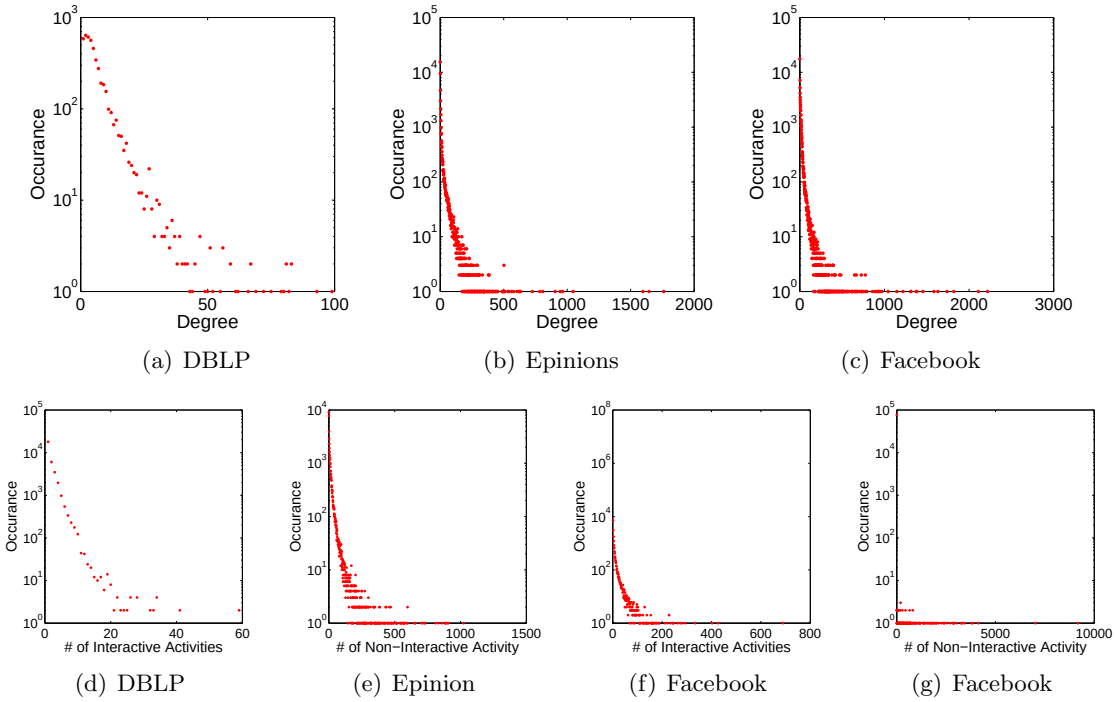


Figure 5: Dataset Properties

activities for both e_1 and e_2.

Epinions dataset, collected by Massa[70], contains consumer reviews and trust networks. Epinions is a platform for people to share their experiences and to maintain a trust network. Customers will be influenced by reviews when they consider buying products. These reviews are displayed after filtered using users' trust network. For example, if users u and v made some reviews and user w likes u's reviews and does not v's reviews, then w creates a trust list by adding u and does a block list by adding v. Now Epinions displays u's reviews first and hides v's reviews for w. Epinions' dataset has 49,289 users, 664,824 reviews, and 487,181 trust statements after 5-week crawl in 2003. Given a user u and her x number of reviews, we construct a vertex u and set x as the number of non-interactive activities for the vertex u. If a user v adds a user u into a trust network, then we create an edge $e(u, v)$.

Facebook is recognized by many as one of the largest social network services. Each user has her profile page where the owner can post photos, videos, and other owner specific information. For each posting, her friends can leave comments. When a user u posts photos, videos, or statuses, we consider it as non-interactive activities. When a user u

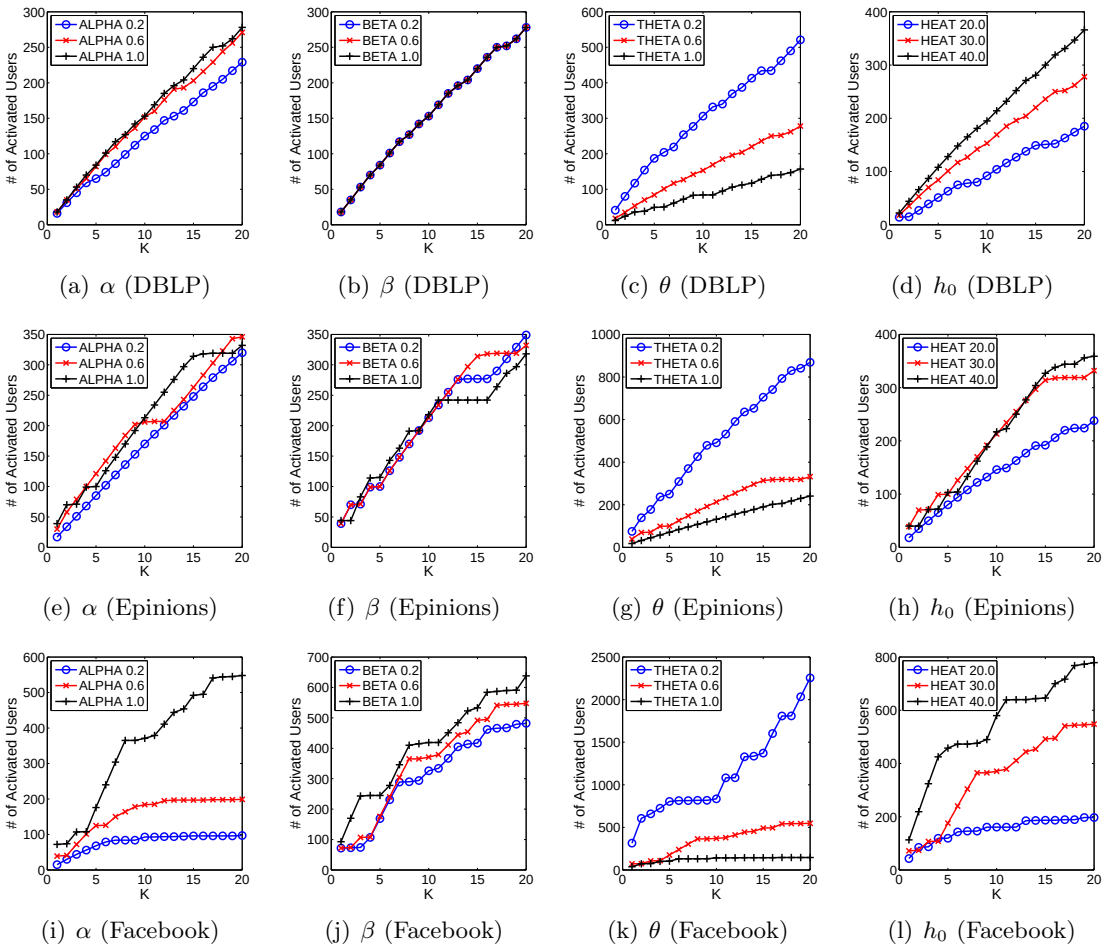
leaves a comment on v's posting, then we consider it as an interactive activity. Given a user u and her x number of postings, we create a vertex u and set x as the number of non-interactive activities. Given a friendship between two users u and v, we create two edges $e_1(u, v)$ and $e_2(v, u)$. If node u has y number of comments on node v's posting, then we create an edge $e(u, v)$ and set y, the number of interactive activities, as the weight for this edge. We launched a Facebook app to analyze users' posting trends and statistics of friends. From 273 Facebook app users, we extract their friend relationship information which creates 76,954 nodes and 1,121,861 friendship edges. For example, one user may have 300 friends. Then we can create 301 users nodes. Repeating this procedure for all 273 users results in 76,954 nodes.

Figure 5 shows distributions of degree, number of non-interactive activities, and number of interactive activities in all three types of datasets. Facebook dataset has both interactive and non-interactive activity information, while DBLP dataset has only interactivity activities and Epinions dataset has only non-interactive activities.

2.5.2 Effect of Various Parameters

In our ActivityInfluence model, we use parameters such as a heat conductivity, α, an acceptance threshold, θ, an activity weight, β, and initial heat, f_0. Different settings of these parameters may significantly affect the heat diffusion process and consequently the influence computation result. Before assigning values for experiments, Figure 6 presents how these values affect heat diffusion process. We initialize α as 1.0, β as 0.6, θ as 0.6, and initial heat h_0 as 30 and measure the number of influenced nodes by varying the value of each parameter.

The heat conductivity, α, controls the speed of heat spread. Some social networks spread news or rumors quickly while others do not. By setting the value of α, we can mimic the speed of spread. The value of *alpha* is a real number between 0 and 1. If it is set to 0, heat will not be diffused at all. On the other hand, if α is set to 1, all of the heat at the heat source will be diffused to its neighbors without any loss. Figures 6(a), 6(e), and 6(i) show how many users are influenced by top-k people while varying α from 0.2 to 1.0. As stated



(a) α (DBLP) (b) β (DBLP) (c) θ (DBLP) (d) h_0 (DBLP)

(e) α (Epinions) (f) β (Epinions) (g) θ (Epinions) (h) h_0 (Epinions)

(i) α (Facebook) (j) β (Facebook) (k) θ (Facebook) (l) h_0 (Facebook)

Figure 6: Effect of Parameters

above, higher α value results in more influenced users because more heat will be diffused.

The activity weight, β, regulates how much weight will be given for non-interactive activities. If it is set to 0, the number of non-interactive activities does not play a role at all on the heat diffusion process. Then each user loses their heat as proportional to time. If β is high, then a user loses heat with regard to her activities in posting. Figures 6(f) and 6(j) show that higher β value results in larger number of influenced users because heat sources will lose their heat much slower, which means the amount of heat to diffuse decreases slower. Figure 6(b) shows that the total number of influenced users does not change significantly while varying β. In the DBLP dataset, each node has no non-interactive activities. It has only interactive activities, which is co-authorship relation. Therefore, β does not affect the

heat diffusion process. Each node loses its heat as the time goes by.

The acceptance threshold, θ, is used to determine if a user is influenced or not. If $f_i(t)$ is greater than or equal to θ_i, then we consider v_i is influenced by heat source users. θ_i value varies from 0 to 1. If it is low, more users will be influenced. Figures 6(c), 6(g), and 6(k) show that when we decrease θ, the number of influenced users increases. Especially, for Facebook dataset, the number of influence users are increased 6 times more when we decrease θ from 0.6 to 0.2.

The next parameter we measure is the initial heat f_0. We varied initial heat, f_0. The more heat that a user is assigned to, the more users are influenced by the top k nodes in Figures 6(d), 6(h), and 6(l).

In the following experiments, we initialize α as 1.0, β as 0.6, θ as 0.6, and h_0 as 30. Twenty most influential people are selected based on three criteria: independent InfluenceCoverage(C1), locally optimal InfluenceCoverage(C2), globally optimal InfluenceCoverage(C3). We compared activity-based heat diffusion with topology-based heat diffusion using three influence ranking criteria: independent InfluenceCoverage(C1), local optimal InfluenceCoverage(C2), and global InfluenceCoverage(C3). For each experiment, we show that global influence ranking is the best top k influence ranking algorithm for selecting the top k nodes that have the overall maximal influence in terms of non-overlapping node coverage.

2.5.3 DBLP Dataset

Experiments on DBLP dataset has display some interesting results. The number of influenced people nodes based on the influence ranks computed by topology-based heat diffusion is larger than thatthe one by activity-based heat diffusion.

DBLP dataset has only interactive activities information. Therefore heat loss rate, β, does not affect the result of heat diffusion. The only difference between topology-based and activity-based approach is the heat distribution. In the topology-based approach, user u_i diffuses its heat evenly to its neighbors. For example, if u_i has 10 co-authors, then

$\frac{1}{10}H_i(t)$ is distributed to each of u_i's neighbors evenly. On the other hand, in the activity-based approach, u_i diffuses its heat to its neighbors based on the rate of interaction. For example, if 90% of interaction is done with u_j, then u_i diffuses $\frac{9}{10}H_i(t)$ to u_j and $\frac{1}{10\times 9}H_i(t)$ is distributed to other 9 neighbors evenly. Therefore, the topology-based heat diffusion has larger number of influenced people as shown in Figure 7(a).

Table 1 shows user node IDs selected as top-20 users. If a user is selected by all three criteria, we highlightshaded the ID inas gray. For example, user 1060 is selected by all three criteria in topology-based approach. Considering the topology-based approach first, from C1 to C2 four users (1422, 4704, 3719, and 3536) are excluded in order to minimize overlap. However, in the activity-based approach, 6 users (4260, 1014, 1012, 4532, 4172, and 3807) are excluded. That is why the number of influenced people by activity-based approach is lower as shown in Figure 7(a).

Experiments on DBLP dataset display some interesting results. The number of influenced nodes based on the influence ranks computed by topology-based heat diffusion is larger than that by activity-based heat diffusion.

DBLP dataset has only interactive activities information. Therefore heat loss rate, β, does not affect the result of heat diffusion. The only difference between topology-based and activity-based approach is the heat distribution. In the topology-based approach, user u_i diffuses its heat evenly to its neighbors. For example, if u_i has 10 co-authors, then $\frac{1}{10}H_i(t)$ is distributed to each of u_i's neighbors evenly. On the other hand, in the activity-based approach, u_i diffuses its heat to its neighbors based on the rate of interaction. For example, if 90% of interaction is done with u_j, then u_i diffuses $\frac{9}{10}H_i(t)$ to u_j and $\frac{1}{10\times 9}H_i(t)$ is distributed to other 9 neighbors evenly. Therefore, the topology-based heat diffusion has larger number of influenced people as shown in Figure 7(a).

Table 1 shows node IDs selected as top-20 users. If a user is selected by all three criteria, we highlight the ID in gray. For example, user 1060 is selected by all three criteria in topology-based approach. Considering the topology-based approach first, from C1 to C2 four users (1422, 4704, 3719, and 3536) are excluded in order to minimize overlap. However, in the activity-based approach, 6 users (4260, 1014, 1012, 4532, 4172, and 3807)

are excluded. That is why the number of influenced people by activity-based approach is lower as shown in Figure 7(a).

We visualized Table 1 in Figures 8 and 9. Figure 8 shows the top 20 nodes using activity based social influence ranking. Figure 9 shows the top 20 nodes using topology-based social influence ranking. Each node is denoted as either a circle, a triangle, a rectangle, or a pentagon. The shape depends on the number of times that the node is selected as one of top-k nodes. If a node is included as one of top-k nodes only once, we denote the node as a triangle. If it is included twice, the shape is rectangle. For a node included three times, its shape is pentagon. For each node v that is considered as one of top-k nodes at least once, then we add their neighbor nodes as a circle. Node identification number is attached to each node. Also we attach the rank of node for each category. For example, node v_{1996} is selected as one of top-k node for three times and is represented as a pentagon. Its rank in selection category C1, C2, and C3 are 1, 1, 1, respectively. On the other hand, node v_{4260} is selected as one of top-k influential nodes only once under the selection category C1. Thus its shape is triangle.

There are 5 pentagons in Figure 9 and 8 pentagons in Figure 8. That is, top-k nodes in topology-based approach are changed more as we change the selection category than activity-based approach. Considering only topology results in selecting top-k nodes with large overlap in InfluenceRank Coverage. Also note that the size of clusters in Figure 9 is much larger than one in Figure 8. For example, there is a huge clusters consisting of 3536, 3622, 1635, 3989, 3719, 1707, 3141. However, Figure 8 has a cluster of size 4, which is 3066, 4532, 4172, 582. Topology-based approach selects nodes that consists clusters, which results in InfluenceRank Coverage overlap.

We also visualize top-20 nodes for topology-based approach and top-20 nodes for activity-based approach at the same time as shown in 14. In order to visualize nodes clearly we only show nodes under the selection category C3. A node has id and two ranks: InfluenceRank in Topology-based approach and InfluenceRank in Activity-based approach. We denote a node as a pentagon if the node is ranked in both approached. Otherwise, the node is represented as a rectangle. There is only one node that is ranked in both approaches, node

1593. Two pairs of nodes are connected each other. Except these four nodes, all other nodes are not directly connected. In DBLP datasets, influential nodes are distributed in different clusters.

2.5.4 Epinions Dataset

Experiments on Epinions dataset show that the activity-based approach has larger influence coverage than the topology-based approach. Epinions dataset has non-interactive activities only. Therefore, users in Epinions dataset diffuse heat evenly to its neighbors like in the topological approach. However, the number of influenced people computed by the activity based heat diffusion approach is different from that by the topology-based approach due to the number of non-interactive activities. Users lose its heat at a slower rate based on the number of non-interactive activities, which is regulated by heat loss factor β. Users with more non-interactive activities lose its heat much slower than the ones with smaller number of non-interactive activities. Thus the number of influenced people by using the activity-based approach is larger than that by the topology-based diffusion as shown in Figure 7(b).

Considering Table 1, selected marketing candidates by two approaches in Epinions dataset are similar while ones in DBLP dataset are quire different. In Epinions dataset, users who has high degree also has high number of non-interactive activities. For example, user u has high number of reviews. Then other users read u's reviews and construct a trust link. Then u's degree increases. In either heat diffusion approach, u diffuses its heat evenly to its neighbors. Thus, if u is selected from the topological approach, then u might be selected because same rate of heat is diffused and the amount of heat is higher.

Figures 12 and 13 show the top 20 nodes using activity based social influence ranking and topology-based social influence ranking over the Epinions dataset respectively. Similar to DBLP datasets, 11 nodes are selected by all three selection categories in Figure 12 but only 9 nodes are selected three times in Figure 13. Activity-based approach selects nodes with less InfluenceCoverage overlap. Note that both approaches have a cluster of size 4, but activity-based approach has only one cluster with 4 nodes and topology-based approach has

two clusters. Thus, topology-based approach has larger overlap in InfluenceCoverage.

Figure 16 shows top-20 nodes for both topology- and activity-based approaches. Unlike other two datasets, we cannot find a link between any two nodes. All nodes are not connected directly. Instead, there are 13 nodes that are ranked in both approaches. Most of them are highly ranked nodes. We believe that Epinions datasets have only NA information and nodes with large NA also have large degree. Thus, two approaches may select similar nodes as influential ones.

2.5.5 Facebook Dataset

Experiments on Facebook dataset show some of the most interesting results. Facebook dataset has both interactive and non-interactive activities. Figure 7(c) shows the experimental results for Facebook datasets.

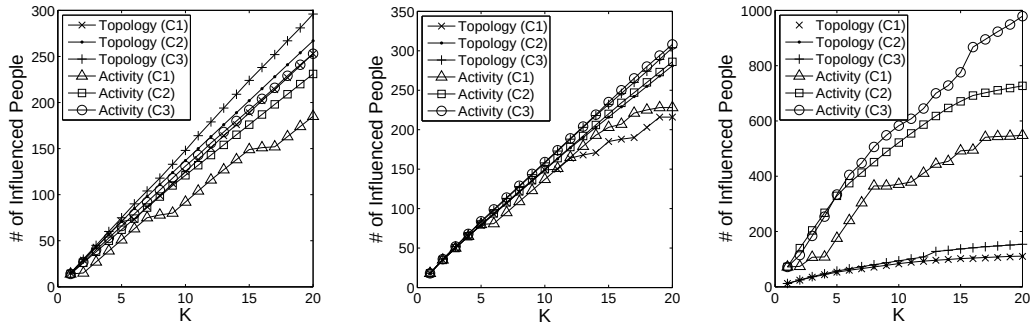
It is interesting to note that for each influence rank based top k selection criteria, the activity-based heat diffusion process influence more users for two reasons. First, in the topology-based approach, users lose heat based on the time period. On the other hand, in the activity-based approach, users loose heat based on the number of non-interactive activities. Thus, some users lose heat quickly while others lose slowly, which means these users have high amount of heat sources to diffuse. Second, users in Facebook dataset have much more non-interactive activities than the ones in Epinions dataset as shown in Figures 5(e) and 5(g). Therefore, the gap in Facebook dataset between the topological and activity-based approach is much higher than that in Epinions dataset.

Unlike other two datasets, Table 1 shows that there are very few users are replaced from one criteria to another criteria in the topology-based approach. For example, from C1 to C2, only one user (180) is excluded in order to minimize overlap. Figures 10 and 11 show that users in the topology-based approach have less overlapping. However, the activity-based approach can effectively replace the number of overlapping users.

Figures 10 and 11 show the top 20 nodes using activity based social influence ranking and topology-based social influence ranking over the Facebook dataset respectively. Facebook data also shows that activity-based approach selects nodes with less overlap. In Figure

10 has 16 pentagons while Figure 11 has only 8 pentagons. Note that topology-based approach has two large clusters while activity-based has small clusters. Due to large clusters, topology-based approach changes the list of top-k nodes as we change the selection category.

We visualize top-20 nodes selected by both approaches in Figure 15. There are two pentagon nodes, which means only two nodes are selected by both approaches. Like DBLP datasets, two approaches barely share influential nodes due to their selection categories. Note that there are five one-hop away clusters. Unlike two datasets, Facebook datasets shows friend relationships. In our datasets, one user has more than 1,000 friends. Among 1,000 friends there are also friend relationships. Thus, we can see more directly connected nodes than other two datasets.



(a) DBLP (b) Epinions (c) Facebook

Figure 7: Accumulated Number of Influenced Users

2.6 Conclusion

We have presented the activity-base social influence model based on activity enhanced heat diffusion kernel and a suite of activity influence rank based top k algorithms. Our ActivityInfluence model has made three unique contributions. First, we introduce a novel mechanism to extend the heat diffusion model by effectively incorporating both interactive and non-interactive activities. Second, we develop a suite of top k influence rank based node selection algorithms by minimizing the overlapping in the node coverage of top k most influential nodes, including independent influence rank, (b) locally optimal influence rank and (c) globally optimal influence rank using Hill Climbing algorithm. Finally we

Table 1: Top-k users for DBLP (DT, DA), Epinions (ET, EA), and Facebook (FT, FA) datasets

	Top-k users									
	1	2	3	4	5	6	7	8	9	10
DT(C1)	1060	4063	3807	2875	2169	1422	993	4704	4466	4390
DT(C2)	1060	4063	3807	2875	2169	993	4466	4390	4129	3989
DT(C3)	1060	993	1422	2875	3807	4063	1205	1207	2338	1707
	11	12	13	14	15	16	17	18	19	20
DT(C1)	4129	3989	3810	3719	3658	3622	3536	3282	3199	3141
DT(C2)	3810	3658	3622	3282	3199	3141	2632	2436	1818	1675
DT(C3)	1624	1660	1542	2424	536	1593	1635	1675	1892	2436
	1	2	3	4	5	6	7	8	9	10
DA(C1)	1966	4260	2345	2303	2085	1593	1495	1014	1012	637
DA(C2)	1966	2345	2303	2085	1593	1495	637	582	553	4756
DA(C3)	1966	553	582	2664	637	1012	1014	1593	3907	2345
	11	12	13	14	15	16	17	18	19	20
DA(C1)	582	553	4756	4665	4577	4532	4172	4156	3907	3807
DA(C2)	4665	4577	4156	3907	3807	3775	3743	3417	3351	3206
DA(C3)	2515	3351	2630	3066	3125	3181	3206	4577	3417	3743
	1	2	3	4	5	6	7	8	9	10
ET(C1)	39245	29998	31672	31267	48488	43253	41947	41554	40305	35168
ET(C2)	39245	29998	31672	31267	48488	43253	41947	41554	40305	35168
ET(C3)	39245	29998	31267	31672	2155	18473	20651	29153	35168	41554
	11	12	13	14	15	16	17	18	19	20
ET(C1)	29153	20651	18473	15219	14006	12596	2155	49080	48519	47211
ET(C2)	20651	14006	49080	48519	45821	43399	43338	40431	39888	39622
ET(C3)	41947	43253	34068	39888	32058	32431	34961	35127	37111	43399
	1	2	3	4	5	6	7	8	9	10
EA(C1)	39245	29998	31672	31267	29153	48488	45872	43338	43253	41947
EA(C2)	39245	29998	31672	31267	29153	45872	43338	43253	41947	41554
EA(C3)	39245	29998	29153	31267	31672	1385	2155	4569	18473	20651
	11	12	13	14	15	16	17	18	19	20
EA(C1)	41554	40305	35168	28707	28049	24983	20651	18473	15219	14006
EA(C2)	40305	35168	28707	20651	4569	1385	49080	48519	48084	45821
EA(C3)	28049	35168	41554	41947	43253	43824	48519	43338	39888	40017
	1	2	3	4	5	6	7	8	9	10
FT(C1)	157	115	172	141	99	188	145	93	83	75
FT(C2)	157	99	188	145	93	200	171	231	10	182
FT(C3)	157	172	115	197	144	83	99	94	171	31
	11	12	13	14	15	16	17	18	19	20
FT(C1)	94	25	182	96	171	180	200	87	31	197
FT(C2)	172	167	25	16	38	58	96	98	94	144
FT(C3)	183	25	96	98	93	143	132	77	151	40
	1	2	3	4	5	6	7	8	9	10
FA(C1)	90	53	206	44	4	193	171	158	101	80
FA(C2)	90	53	206	44	4	193	158	101	80	58
FA(C3)	90	53	206	4	44	193	164	58	80	101
	11	12	13	14	15	16	17	18	19	20
FA(C1)	58	191	49	236	172	180	160	98	37	9
FA(C2)	171	49	191	236	172	160	98	37	9	207
FA(C3)	158	171	180	49	191	172	236	40	9	37

conduct an extensive series of experiments on three representative real-world social network datasets to show the effectiveness of our activity-based social influence model and influence rank algorithms. Compare to the existing topology-based influence diffusion model, the activity-based social influence model considers not only topology of a social network but also activity sensitive attributes such as interactive activities, non-interactive activities and time stamps. Our ongoing research continues along two dimensions. First, we are interested in further investigating other types of social network attributes that are critical to social network analysis, such as time stamps and similarity of user profiles. Second, we are also interested in studying different types of incentives and how they impact on the social influence and the spread of information over a given social network.

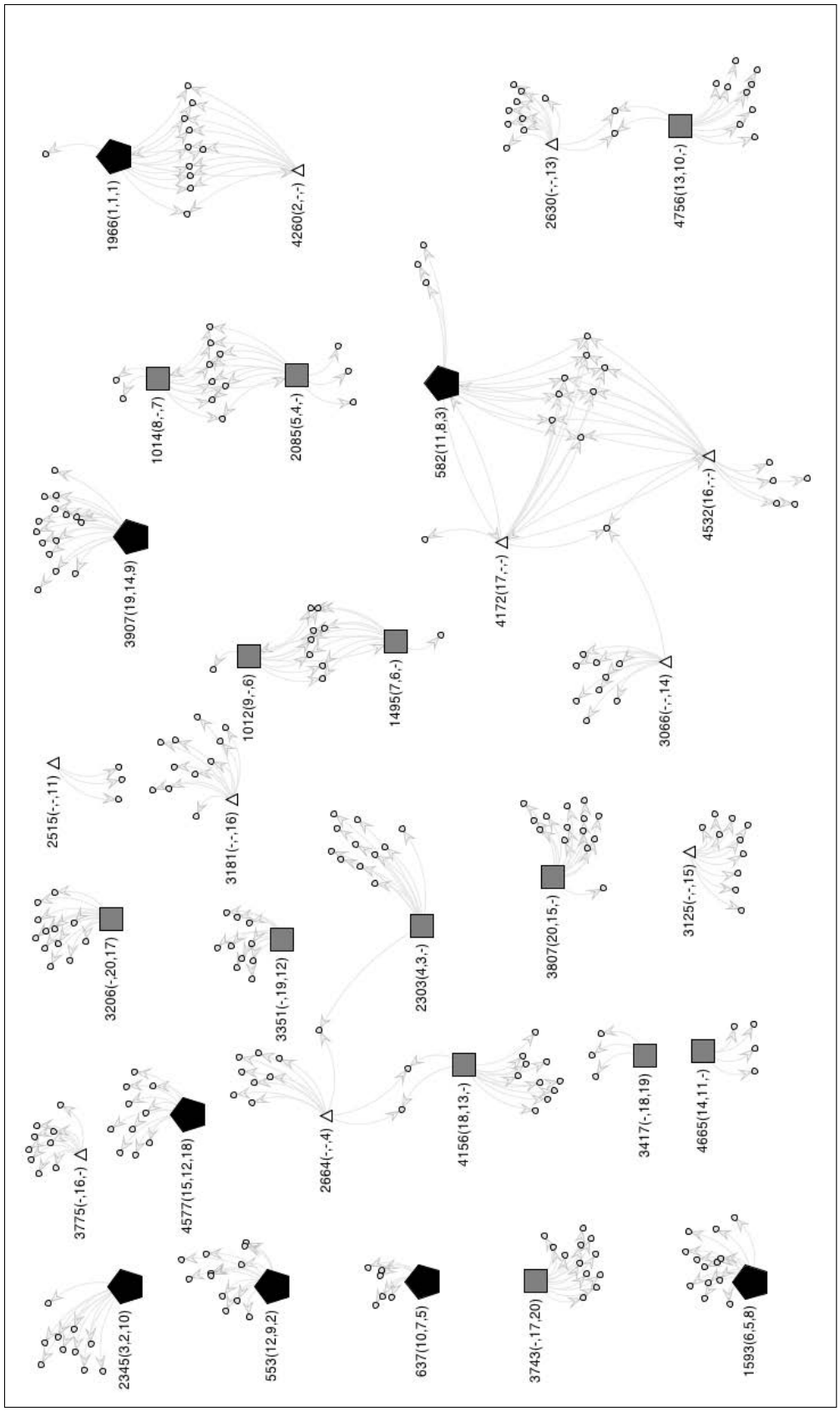


Figure 8: Top-20 nodes by Activity approach(DBLP)

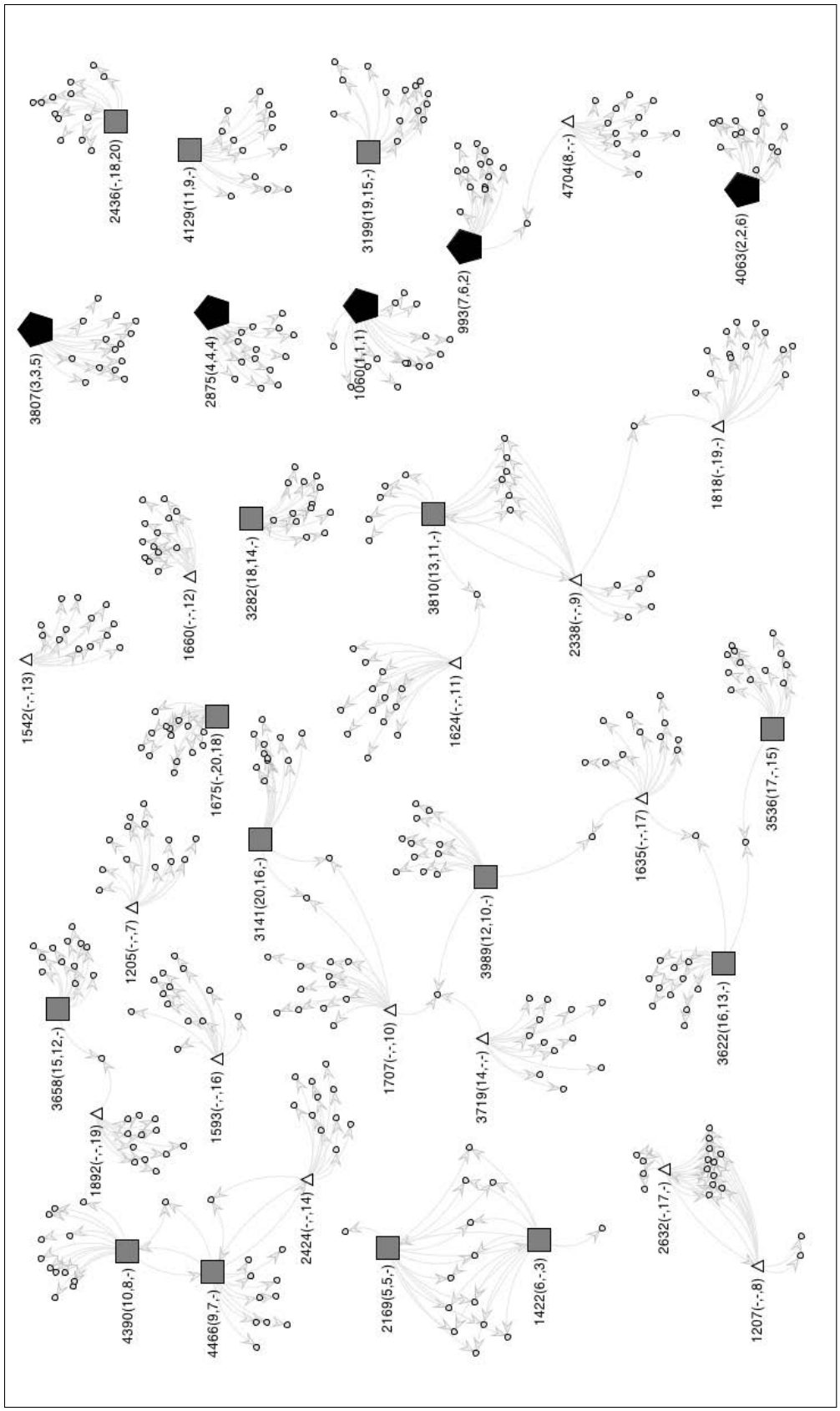


Figure 9: Top-20 nodes by Topology approach(DBLP)

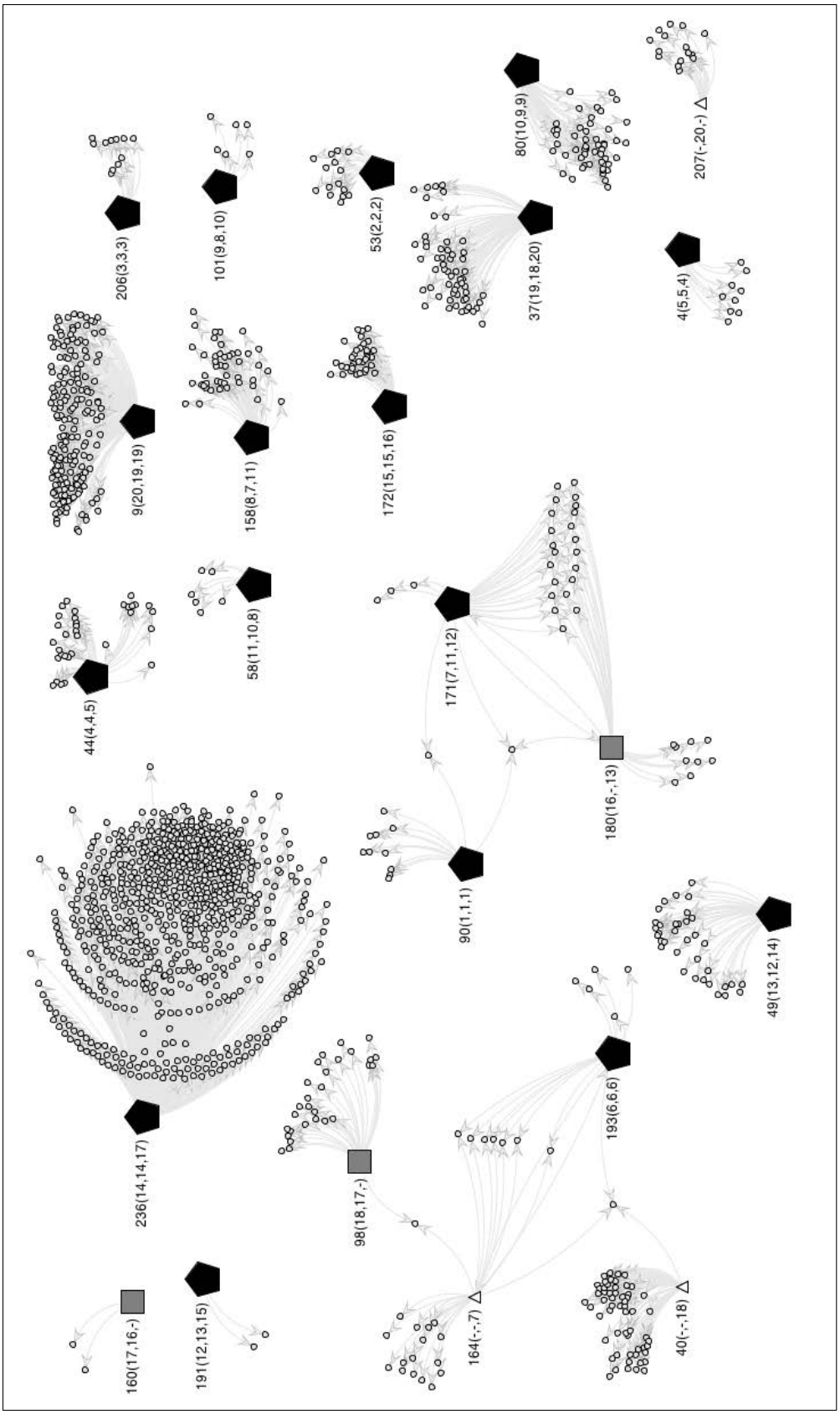


Figure 10: Top-20 nodes by Activity approach(Facebook)

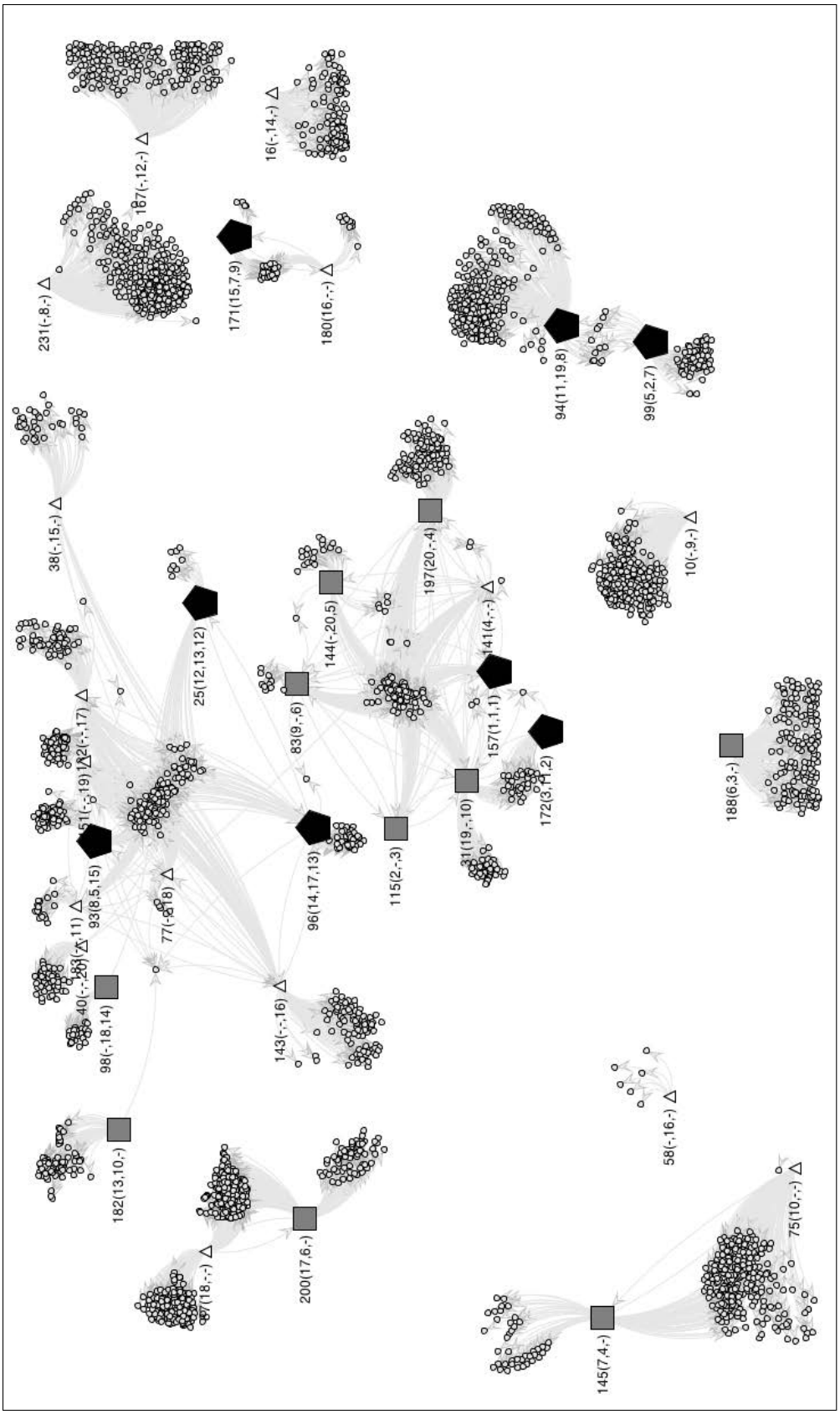


Figure 11: Top-20 nodes by Topology approach(Facebook)

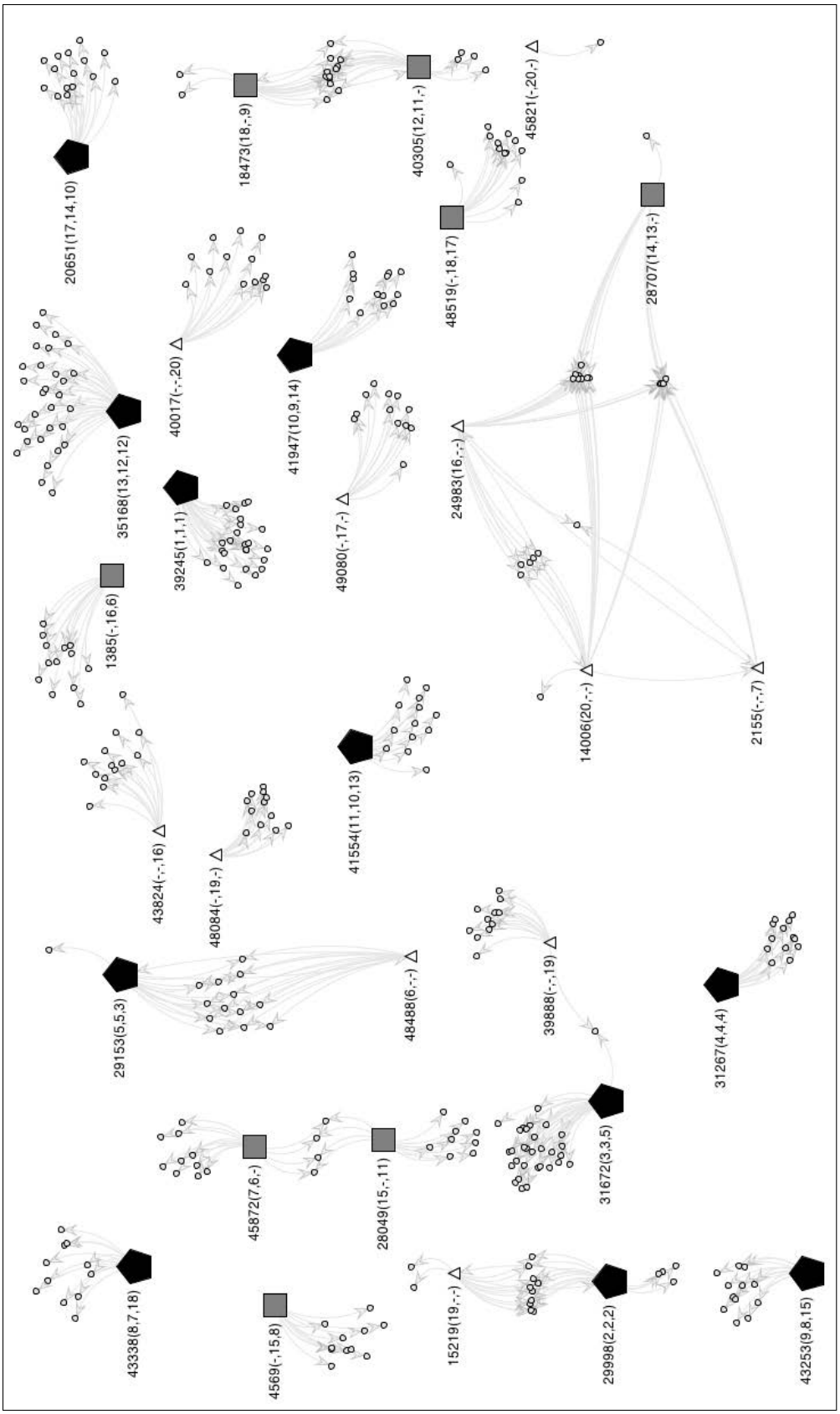


Figure 12: Top-20 nodes by Activity approach(Epinions)

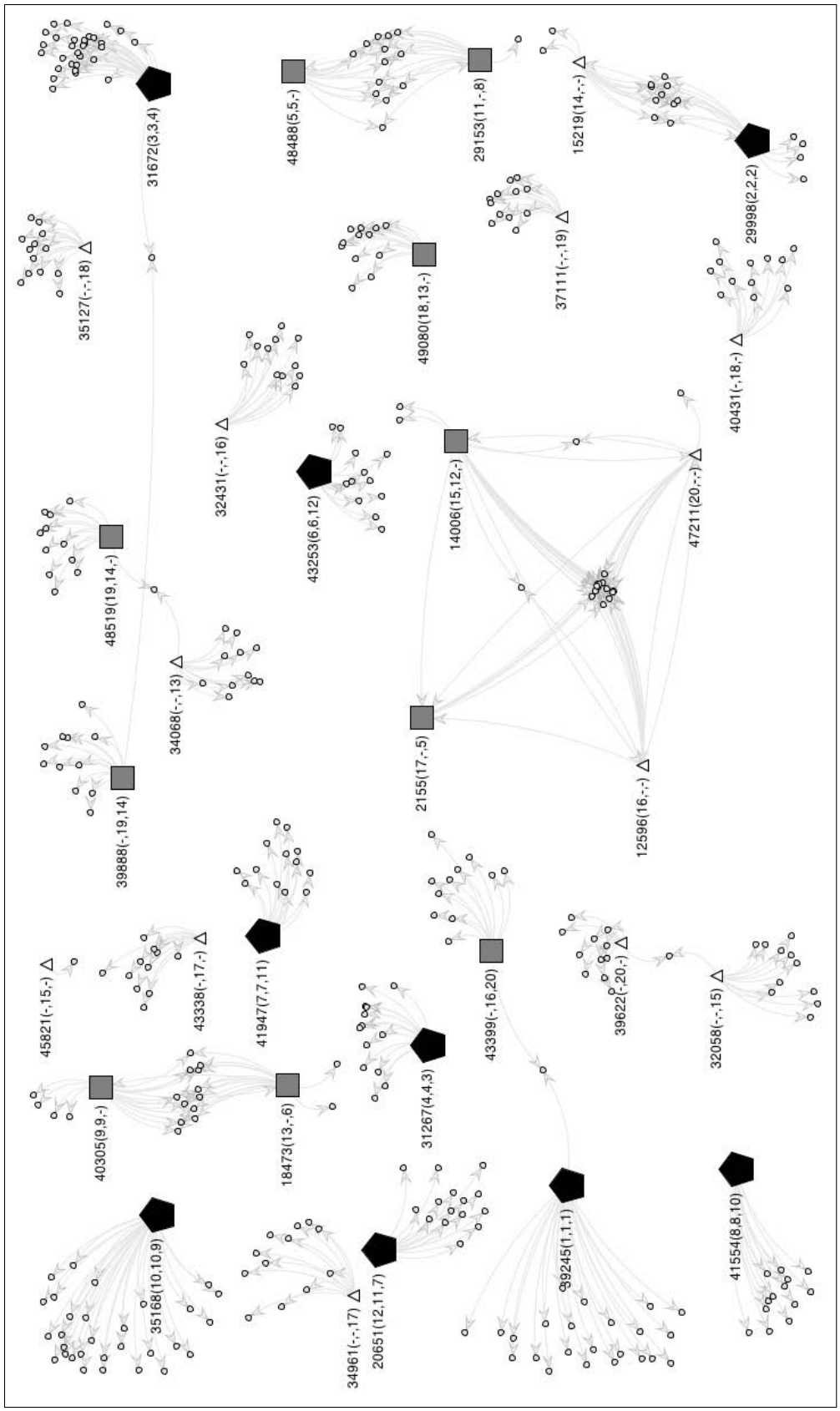


Figure 13: Top-20 nodes by Topology approach(Epinions)

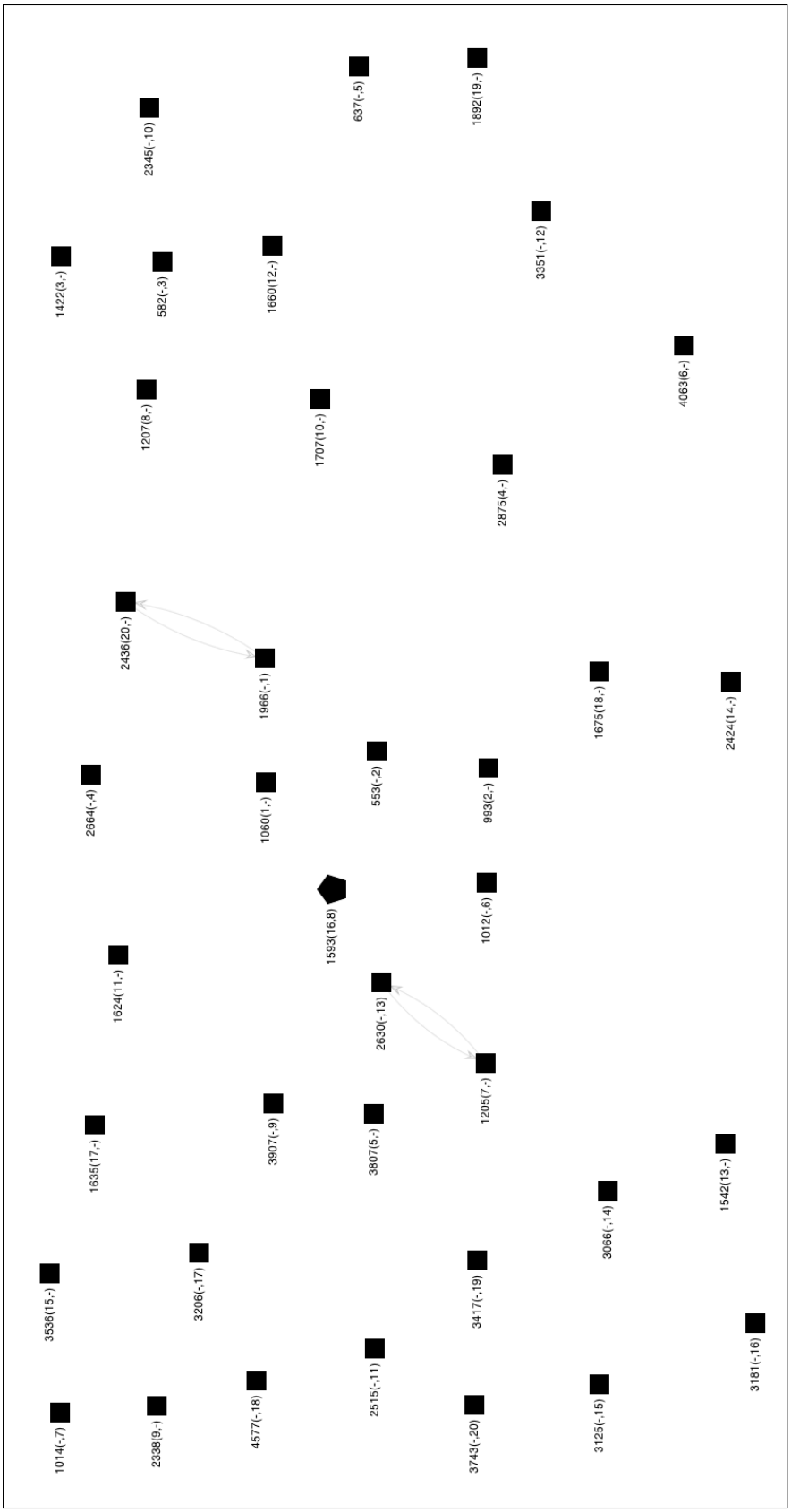


Figure 14: Top-20 nodes Comparison (DBLP)

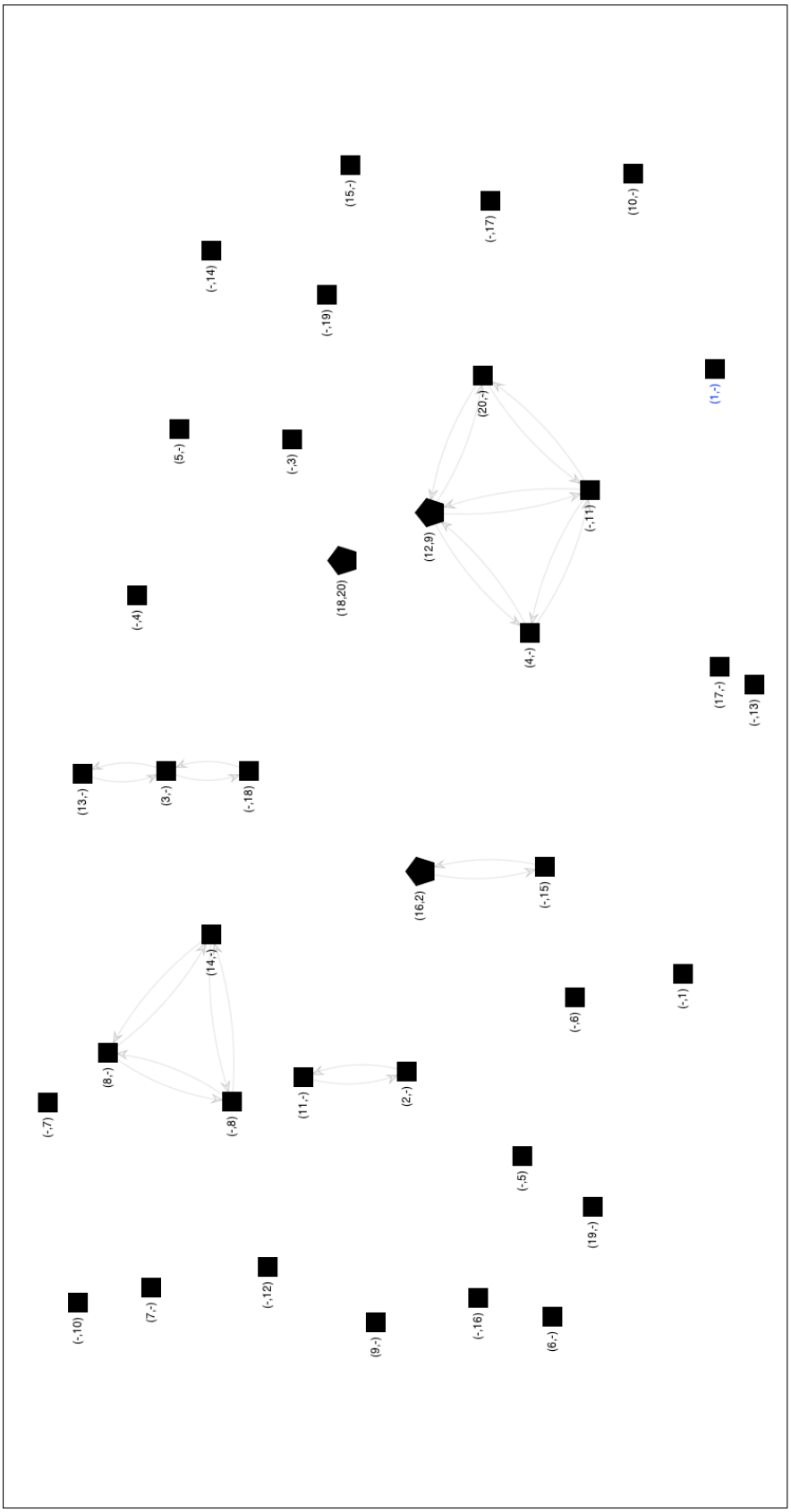


Figure 15: Top-20 nodes Comparison (Facebook)

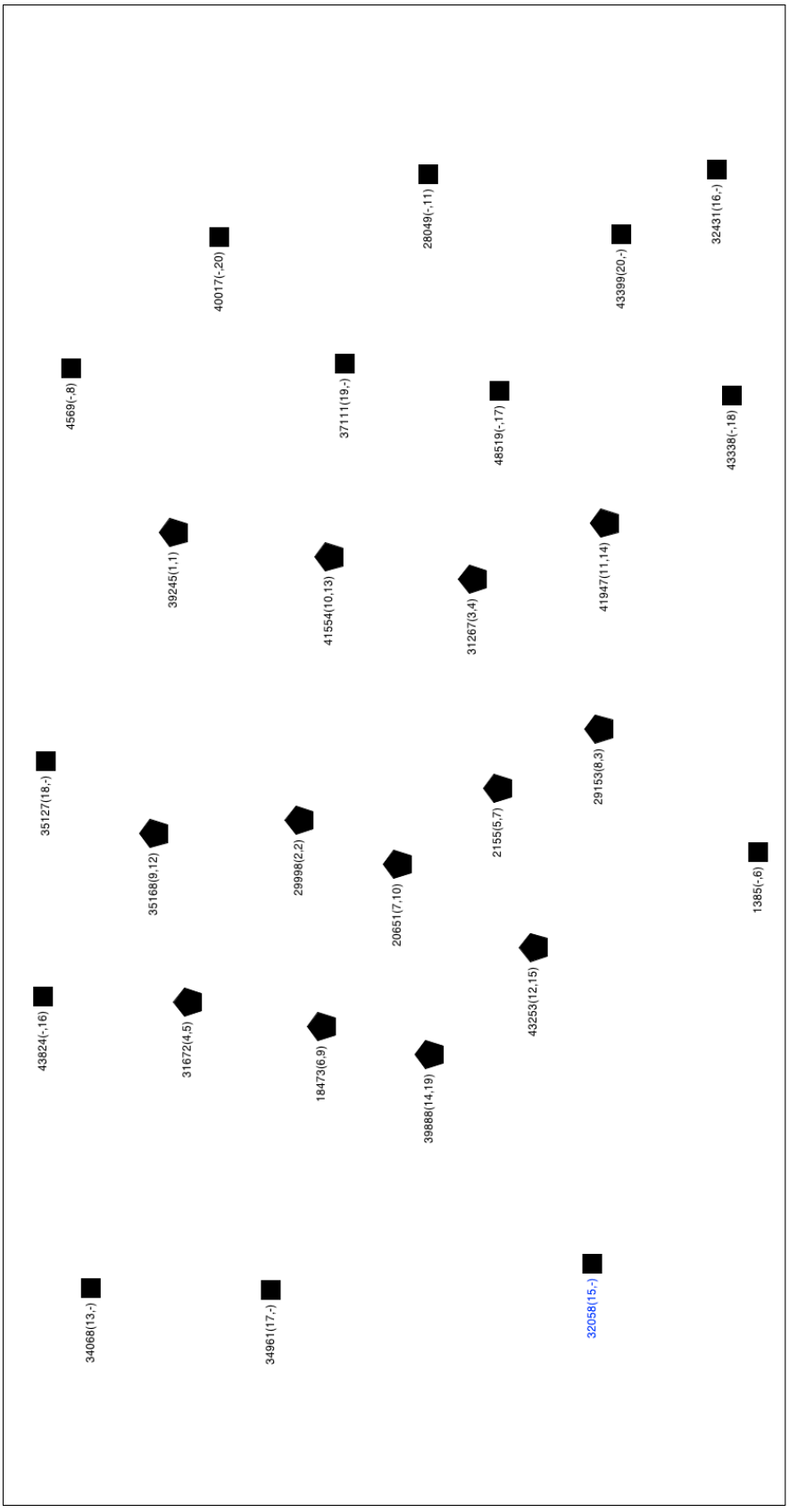


Figure 16: Top-20 nodes Comparison (Epinions)

CHAPTER III

PROBABILISTIC DIFFUSION OF SOCIAL INFLUENCE WITH INCENTIVES

3.1 Introduction

Social network analysis centers on the study of the roles which the graph of relationships and interactions among a group of individuals plays in the diffusion of influence and spreading of information among the members of the group. There are alternative models for simulating how the ideas and influences are diffused and propagated through a social network. For example, the diffusion of medical innovation, or the sudden spread of viruses and contagious diseases have been studied in bioinformatics and health science domain. The effect of "word of mouth" has been studied in business marketing, and the pollution propagation in the water networks has been studied in technological settings.

In Chapter 2, we have shown the advantages of using heat diffusion kernel to model both topology-based social influence diffusion and activity-based social influence diffusion. Although the heat diffusion kernel is simple and straightforward in capturing the basic principle of social influence among a social network of people, there are several serious limitations of using heat diffusion to model social influence among people. In this chapter we first analyze the inherent problems with modeling social influence as heat diffusion process and then explore alternative models that are more effective for modeling social influence diffusion.

The first limitation of using heat diffusion kernel for modeling social influence is the uniform distribution assumption. In the heat diffusion based activity influence model, every node participates in the heat diffusion process following two dimensions. On one hand, a node receives heat from its neighbor nodes that have higher heat values, and on the other hand, it propagates its heat to those neighbor nodes with lower heat values. This diffusion process continues to iterate through every node in the network until the heat diffusion

process converges. An example convergence condition is that there is no significant heat change at a majority of the nodes. However, the composition of online social networks is primarily individuals with different mindsets and different beliefs. Not all friends of a node are equally interested in taking a part in the influence diffusion process at all types of information contexts. It is highly likely that some friends of a node are more actively engaged in the diffusion process whereas the others are relatively speaking less interested or engaged in the influence diffusion process. The former can be viewed as a spreader and the latter can be viewed as a stiffer or a stopper depending on the degree of indifference. We argue that the influence diffusion from a node u to a neighbor node v is highly probabilistic and highly selective.

The second limitation of heat diffusion model is the assumption that high degree nodes are eigher better heat sources or more active in heat diffusion process. We observe that nodes that are active in social influence diffusion may not always be the nodes with high degree in the social network. Furthermore, nodes that are active in the first round of influence diffusion may not want to participate in the diffusion of the same information. We argue that a probabilistic approach to social influence diffusion may be more realistic and practical.

Third but not the least, in the heat diffusion kernel for a social network of size N, the diffusion process runs for a total of N rounds or until the influence diffusion converges. In each round, direct links (such as friendship) are used to propagate information and influence from one node to another node in the form of "word-of-mouth" communication. According to [28], by analyzing 5.2 billion twitter friendships and discovered that on average Twitter is a network with only 5 degree of separation. This implies that users in Twitter are five hops away from each other. This understanding confirms the Milgram's [88] small world experiment. In Milgram's experiment, a sample of US individuals were asked to reach a particular target person by passing a message along a chain of acquaintances. The average length of successful chains turned out to be about five intermediaries or six separation steps (the majority of chains in that study actually failed to complete). A recent electronic small world experiment at Columbia University [90] found that about five to seven degrees

of separation are sufficient for connecting any two people through email. We can view the small world phenomenon as the hypothesis that the chain of social acquaintances required to connect one arbitrary person to another arbitrary person anywhere in the world is generally short. The concept gave rise to the famous principle of six degree of separation, which implies that the information can be propagated faster in social networks. Thus it is critical to identify the critical convergence conditions for each given diffusion model.

Finally, we argue that the heat diffusion based social influence model is not suitable to study whether and how incentives may stimulate the rate and the coverage of influence diffusion in a social network. Incentives often play a critical role in creating stimuli for spreading of information or innovation and diffusion of influence within a short time window to a broader coverage of the network.

Bearing these issues in mind, in this chapter we propose a probabilistic approach to modeling social influence diffusion model with multi-scale rewards as incentives. Our probabilistic diffusion approach has three unique features. First we define an influence diffusion probability for each pair of nodes, which can differentiate nodes that have higher level of interactions and higher node degree from those nodes that have low or zero interactive activities. Second, we categorize that nodes in the social network graph into two classes: active and inactive nodes. Active nodes can have one chance to influence inactive nodes but not vice versa. Third but not the least, we utilizes a system defined diffusion threshold, combined with the pair-wise diffusion probability, to control and manage how influence is propagated across the social network of N nodes. When this threshold is set to a high value, the number of nodes that are influenced by a given influence source (node) will be small. By using this probabilistic diffusion model, we can formally study how incentives can be utilized as stimuli to further boost the influence diffusion rate and coverage. For each node in a social network, we compute its probability-based social influence ranking score, which is measured by the approximate node coverage of the influence diffusion initizited from this node. Our experiments show that the reward-powered social influence model is more effective in terms of both diffusion rate and diffusion coverage of influence.

3.2 Problem Statement and Design Overview

In this section we briefly review the main components of two representative influence diffusion models: Heat diffusion based social influence models and stochastic social influence models. We analyze the potential problems with each model and design a probabilistic influence diffusion model with multi-scale reward points as incentives.

3.2.1 Heat Diffusion based Social Influence

We have studied two different heat diffusion models in Chapter 2: topology-based heat diffusion model and activity-based heat diffusion model. The former defines the heat diffusion process in terms of topological structure of the social network. Influential nodes (people) are often high degree nodes with many friends or followers, while less influential nodes tend to have a small number of friends or followers. The latter defines heat diffusion process in terms of activities performed at each node and between a pair of nodes. Thus the most influential nodes tend to have both high degree and high level of activities. The activity-based heat diffusion model enhances the topology based diffusion model by taking into account of two types of activities (interactive and non-interactive) and by promoting the observation that nodes that engage in more activities tend to have higher influences on its direct friends and its circle of friends (friends of friends).

Let $G =< V, E >$ denote a social network of N nodes with $V = \{v_1, v_2, \ldots, v_N\}$. We say that node v_j is a direct neighbor of node v_i if (v_i, v_j) is in E. The heat diffusion based influence propagation is performed as follows:

- Step 1: For each node $v_i \in V$, assign $H(0)$ amount of heat as its initial heat value.
- Step 2: Simulate the diffusion of influence of v_i to all of its friends using the heat diffusion kernel (e.g., Eq. (10) in Chapter 2) for a period of time t. In the topological heat diffusion model, the amount of heat to diffuse is proportional to the degree of the node u, denoted by $\frac{1}{d_u}$, where d_u is a degree of node u. In the activity based heat diffusion model, the amount of heat to diffusion is proportional to not only the degree of u but also the amount of interactive activities u has with its neighbors, denoted

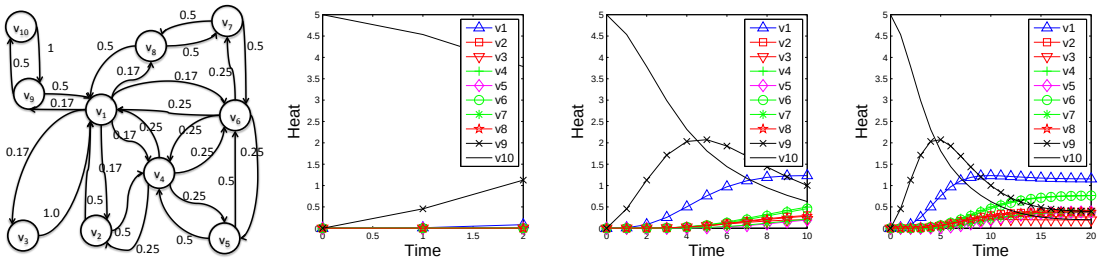
as $IA(u)$ and the amount of non-interactive activities u has performed, denoted as $NA(u)$. Let N denote the size of G and β denote a system-defined weight to the amount of heat preserved at a node during the diffusion process. We can compute the amount of heat diffused from v_i to v_j, denoted by $H(i,j)$ using Eq. (18):

$$K(i,j) = \begin{cases} \frac{IA_{ji}}{\sum_{k:(v_j,v_k)\in E} IA_{jk}} & (v_j, v_i) \in E \\ -1 + \beta \frac{NA_i}{MAX_{1\leq j\leq N}(NA_j)} & i = j \text{ and } d_i > 0 \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

- Step 3: Count the number of nodes whose heat is greater than or equal to the system defined diffusion threshold θ. We call this count value InfluenceCoverage of v_i.

Given that heat continues to diffuse, at any given time t, we can use Eq. (10) (recall Chapter 2) to compute the amount of heat at each node. Figures 17(b), 17(c), and 17(d) show the amount of heat after t period of diffusion from v_{10} on a social network given in Figure 17(a). Color of lines shows the distance of nodes between v_{10} and others. Black indicates that nodes are 1-hop away, red shows that nodes are 2-hop away, green gives the nodes 3-hop away, and blue displays the nodes 4-hop away. These three experiments show that if we set t too small, then it is too short to diffuse heat to a larger number of nodes. Thus it is difficult to determine the influence of v_{10} as shown in Figure 17(b). Heat source (node v_{10}) is decreasing its heat but heat values only increase at nodes on-hop away from the heat source. However, if we set t too large, then the heat diffusion process takes too long, more nodes will enter the convergence state and thus have the same heat as shown in Figure 17(d). Nodes that are more than 2-hop away are all entered convergence state around time $t = 10$. Figure 17(c) shows the setting of t ranging from 1 to 10. We can see that for nodes one-hop away from the heat source, their heat values increases in the beginning and starts decreasing at $t = 5$. For nodes 4-hop away from the heat source, their heat values continue to increase until $t = 10$. For nodes that are 2-hop or 3-hop away from the heat source, their heat values increase slowly and gradually until $t = 10$ due to the topology. These experiments show that an important challenge for tuning the heat diffusion

model is how to determine what the right number of rounds of simulation (t) is and when is the best to stop the diffusion process.



(a) Graph (b) Too small t (c) Reasonable t (d) Too large t

Figure 17: Determine t

Another inherent issue in the heat diffusion model is that everyone participates in the influence diffusion process in a deterministic manner. For example, in the topological heat diffusion model, $H(0)$ amount of heat is diffused to all of v_i's friends equally by dividing the heat of v_i to d_i shares and d_i is the node degree of v_i. Similarly, in activity-based heat diffusion model, $H(0)$ amount of heat is diffused to all of v_i's friends based on the set of interactions performed between v_i and other friends of v_i. This assumption is unrealistic since given any node v_i, not all of v_i's friends will be influenced by v_i equally at all times. In the real world, people are influenced more by their close friends and friends with which v_i has frequent interactions. For those friends that v_i has not had active interaction for a long time due to geographical distance or other practical reasons, v_i's influence on them is fairly limited and sometimes quite random.

Based on our experience and observation with the heat diffusion based social influence model, we argue that it is equally important to study the stochastic influence diffusion models and to understand the non-deterministic nature of the social influence diffusion and the role of probability and incentives can be utilized to model the rate and coverage of social influence under the non-deterministic assumption.

3.2.2 Stochastic Influence Diffusion Models

In contrast to heat diffusion model which is deterministic in nature, the stochastic diffusion process involves some indeterminacy: even if the initial condition (or starting point) is

known, there are several (often infinitely many) directions in which the heat diffusion process may evolve. Familiar examples of processes modeled as stochastic time series include stock market and exchange rate fluctuations, signals such as speech, audio and video, medical data such as a patient's EKG (Electrocardiogram) , EEG (Electroencephalography), blood pressure or temperature, and random movement such as random walks.

The two basic Stochastic models that are used to gauge social influence are Independent Cascading model and Linear Threshold model [42]. Both models use a directed graph $G = (V, E)$, where V is a set of vertices representing users and E is a set of edge expressing friendship relationship. In the independent cascade model each node is assigned a random probability to activate its inactive neighbor nodes. If a uniform random number generated for each node is higher than the probability of an active node, it can activate its inactive neighbor nodes. In the linear threshold model, a threshold θ_v is assigned to each node v and an active node needs to compare the total weight of its neighbors against the threshold of v to determine whether the information is diffused to its neighbor nodes.

A node u is called *active* if u adopts an innovation, otherwise it is *inactive*. Initially all nodes are inactive. By choosing a node to be an adopter, it becomes active. Thus, social influence of u is computed by counting on the number of nodes that are activated by u.

3.2.3 Independent Cascading Model(IC)

IC [42, 43] takes advantages of user interaction. Each relationship represented by $E(u, v)$ has a probability for u to activate v, denoted by $P_p u, v$. When a node u first becomes active in step t, it is given a single chance to activate its inactive neighbor v with probability $p_{u,v}$ in step $t + 1$. If v is activated by u in step $t + 1$, then v also has one chance to activate its inactive neighbors in step $t + 2$. We start this process with an initial set of active nodes, denoted as A_0. This is an iterative process and it stops when there is no more activation are possible. In [42, 43], $p_{u,v}$ os set by the system randomly. We argue that it is important to utilize the attributes of social network nodes, which are critical to influence diffusion, to define the computation of the probability $P_p u, v$.

Similar to the heat diffusion model, IC also tries to activate all of neighbors regardless

of its closeness.

3.2.4 Linear Threshold Model (LC)

LC [44, 83] considers that each node's tendency to be active increases monotonically as more of its neighbors become active. For example, the more friends of Alice buy new iPhones, the higher desire Alice will have for buying an iPHone. Thus, at some point, v's active neighbors may reach to a level of influence that can trigger an inactive neighbor of v, say u, to be active. This is called node-specific *threshold*.

A node v is influenced by each neighbor u according to $w_{u,v}$ such that $\sum_{v \in E(u,v)} w_{u,v} \leq 1$, where $0 \leq w_{u,v} \leq 1$. In step 0, all nodes are inactive and one node u is active. We activate node v whose sum of $w_{u,v}$ is greater than or equal to θ_v and v is one of u's neighbors. This can be re-written as follows: $\sum_{v \in E(u,v)} w_{u,v} \geq \theta_v$

Thus, θ_v is the key parameter in the LC model. If θ_v is set to be low, then the tendency of activation is high. For example, a new iPhone4S would have lower θ_v than the first generation of iPhone.

Although LT takes advantage of tendency of information propagation, it fails to consider the closeness between a pair of nodes. Furthermore, we argue that it is important to define a meaningful node threshold to replace randomly generated threshold based on node degree, amount of activities and the rate of the information to be diffused across the network.

In summary, we argue that a probabilistic influence diffusion approach powered with rewards will outperform activity based or topology based heat diffusion model.

3.3 The Probabilistic Social Influence Model (PSI)

We present the basic design of our probabilistic social influence model, which is designed by combining the best of both IC and LT while removing the limitation of each. Comparing with the heat-diffusion model, our PSI model has introduced probabilistic instead of deterministic management of influence diffusion along three dimensions.

First, we argue that the edge weight between a pair of nodes u and v should be probabilistic in nature but the probability should not be randomly assigned. Instead, the probability

of the edge weight should reflect the dynamic interactions between u and v such that the more interactive activities from u to v should result in the higher weight on the edge from u to v and thus the higher probability of u diffusing its influence to v.

Formally, given a social network graph $G = (V, E)$ where V is a set of nodes and E denotes a set of directed edges. Each node $u \in V$ has node attributes, such as the number of non-interactive activities, $NA(u)$. Nodes are either inactive or active. Initially all nodes are inactive. If a node u has performed some interactive activities with v, then an edge $E(u, v)$ is created from u to v with the edge weight defined by the number of interactive activities from u to v, $IA(u, v)$. Based on this collection of information we compute the probability, $w(u, v)$, for u to activate or influence v. We will explain how to compute the probability in the next section

Second, we need to incorporate the probabilistic differentiation factor into our PSI model in order to determine whether and when to stop propagating information. In the real word, we witness at least three categories of social network participants. Some people are really active in posting reviews about new products or new ideas, and promote others to adopt them and/or disseminate them to more people. Some people are only interested in propagating information to their close friends and receive influence from their close friends. Some people are passive participants in the social network and they may read reviews only and do not propagate the information to their friends. Unfortunately, the heat diffusion based influence model establishes the influence diffusion process by assuming every node to propagate information to their friends, which is unrealistic. Thus, in our probabilistic social influence model, we would like to model different types of participants in a social network in terms of their influence diffusion adoption style, such as active, friend only, or non-active) by introducing two parameters: $theta_c$ as the closeness threshold and $A(u)$ as the influence adopter category. We use $A(u)$ to determine how active u is in propagating information and diffusing influence to its friends. Once u decides to propagate information, we use $theta_c$ to determine which friends of u should be encouraged to accept the information or influence. We will describe in detail how to compute and set these two parameters, θ_c and $A(u)$, in Section 3.3.2 and Section 3.3.4 respectively.

Third but not the least, we introduce incentives as a way to stimulate and encourage inactive nodes to become interested and actively engaged in the diffusion process of their neighboring nodes. We will present the probabilistic social influence model with rewards as incentives in Section 3.3.5.

3.3.1 Probability to Influence Friends, $w(u, v)$

When node u first becomes active in step t, u has one chance to activate all of u's inactive friends, say v, with the probability $w(u, v)$. The result of activation is either "active" or "inactive". We can view the outcome of this random event as being determined by flipping a coin of bias probability $w(u, v)$: "head (active)" or "tail (inactive)". In a coin toss, the probability is 0.5 for each case. But in our PSI model, the probability of "activation" is defined by $w(u, v)$ and the probability for "being inactive" is $1 - w(u, v)$.

The probability $w(u, v)$ is computed based on graph attributes. The computed probability, $w(u, v)$, is assigned to the edge $E(u, v)$. $w(u, v)$ can be computed in various ways. For topology-based approach, the probability for u to activate all of its friends are equal and we can only use the degree of node u, d_u. Thus, $w(u, v)$ for all of u's friends is $\frac{1}{d_u}$. However, it is unfair to have the same probability for all of u's friends. Some friends are more likely to be influenced while others may be too stubborn to be influenced.

In our approach we take advantage of activity information such as $NA(u)$ and $IA(u, v)$ for computing $w(u, v)$. $NA(u)$ is the number of non-interactive activities performed by u. Examples of NA are posting reviews about the newly purchased camera or tips for programming shell scripts. Thus we consider that $NA(u)$ represents how actively u generates information that can influence others. This activity is exposed to all of u's friends equally.

$IA(u, v)$ is the number of interaction between u and v. $NA(u)$ is open to all of u's friends. But, $IA(u, v)$ is exclusively dedicated to a pair of users u and v. Therefore the impact of $IA(u, v)$ is limited to v only. To combine these two attributes, we formulate the probability $w(u, v)$ as follows:

$$w(u, v) = \alpha \frac{NA(u)}{MAX(NA)} + (1 - \alpha) \frac{IA(u, v)}{\sum_{s:(u,s)\in E} IA(u, s)} \tag{19}$$

where $\alpha \in [0, 1]$ is a weight function for balancing between non-interactive activities and interactive activities in computing the edge weight probability $w(u, v)$, and $MAX(NA) = MAX_{v \in V}(NA(v))$.

Note that $w(u, v)$ is high when $NA(u)$ and $IA(u, v)$ are relatively large. When α is set to a small value, approaching zero, then large $NA(u)$ will no longer imply high $w(u, v)$ if $IA(u, v)$ is relatively small. Similarly, when α is set to a value approaching one, then large $IA(u, v)$ will no longer imply high $w(u, v)$ unless u has significantly high $NA(u)$.

3.3.2 Closeness Threshold, θ_c

One limitation in the heat diffusion based influence models is the assumption that a node u will activate all of its friends in each diffusion step, regardless of the different level of closeness that u may have with each of its friends. In order to differentiate friends of u who are close, friends of u who are simply acquaintances in the past, or friends of u who are no longer in close contact for years, we introduce a system defined parameter, called the closeness threshold θ_c. In our PSI model, each node u is assigned to a closeness threshold $\theta_c(u)$. u may activate its friends v with the activation probability $w(u, v)$ if and only if the following condition is satisfied: $w(u, v) > \theta_c(u)$. This condition implies that u can only activate or influence v if the activation probability $w(u, v)$ is above u's closeness threshold $\theta_c(u)$. If u is only interested in diffusing influence to its close friend, u can set its closeness threshold $\theta_c(u)$ to be higher. On the other hand, if u is actively engaged in diffusion of influence across the network, then $\theta_c(u)$ should be set to a low value.

For example if we set $\theta_c = 0.3$ for all nodes, u has two friends s and t, $w(u, s) = 0.25$, and $w(u, t) = 0.5$. Thus, u can only activate, or diffuse its influence to s. This is because by $w(u, s) < \theta_c$ even though s is a friend of u but based on the interactive activities s has with u, s is not considered as a close friend of v at the diffusion time, thus the influence diffusion from u to s is not successful by the current activation probability.

Alternatively, we can also relax the closeness parameter by using the aggregate edge weights, $\sum_{v \in E(u,v)} w_{u,v} \geq \theta_c$, to replace the condition of $w(u, v) < \theta_c$. This alternative condition implies that v can be activated b u if the sum of the edge weights from vs friends

to v is higher than the closeness threshold set at u. This implies that u can influence v if v has many highly active friends.

3.3.3 Activation Result, $X_{u,v}$

In the first development of our PSI model, the result of activation, denoted by $X_{u,v}$, is binary mode of either "active" or "inactive". This is the same as a discrete Bernoulli random variable which outputs only a "head (active)" with probability $w(u,v)$ or "failure (inactive)" with probability $1 - w(u,v)$. We can formally define $X_{u,v}$ as :

$$X_{u,v} = \begin{cases} 1, & \text{succeed in activating (head)} \\ 0, & \text{fail to activate (tail), } u = v, \text{ or } (u,v) \notin E \end{cases} \tag{20}$$

and

$$P_X(x) = \begin{cases} w_{u,v}, & x \text{ head} \\ 1 - w_{u,v}, & x \text{ tail} \end{cases} \tag{21}$$

.

3.3.4 Adoption Probability Group, $A(u)$

In order to incorporate different types of nodes in terms of their influence diffusion behavior, we need to classify social network nodes according to their adoption intent with respect to social influence diffusion. According to [82], people can be classified into five groups with respect to their willingness to adopt an innovation: Innovators, Early Adopters, Early Majority, Late Majority, and Laggards. Innovators are people who adopt an innovation for the first among their friends. They are very social and have interaction with other innovators. In [82] it estimates that they are approximately 2.5% of the entire users. They propagate innovation to early adopters who are representing the next 13.5% of the population. Early adopters are the second fastest people who adopt an innovation. They are the next 34% and are more socially forward than early majority. Early Majority adopts an innovation after a varying length of time. Also the time of adoption is significantly longer than the innovators and early adopters. Early Majority tends to be slower in the adoption process, and seldom hold positions of opinion leadership in a system. Early majority propagate innovation to late majority. Late Majority are typically skeptical about an innovation and

very little opinion leadership. The last group is laggards. People in this group are the last to adopt an innovation. Unlike some of the previous categories, individuals in this category show little to no opinion leadership in influence diffusion. Laggards typically tend to be in contact with only family and close friends.

In our PSI model, we adopt the five categories of people proposed in [82] to capture the difference of the social network nodes in terms of their interests and willingness to propagate information. For example, innovators are trend leaders. They adopt and also propagate new ideas and innovation to others actively. Thus, we set high percentage (say 90%) of them to activate others and only a small percentage (say 10%) of them may stop propagation. The percentage of stopper increases as we move to the next category. For the laggards, they are neither into adopting new things (accepting influence) nor propagating it (diffusing influence). Thus we can say that high percentage (e.g. 90%) of them are classified as stoppers. Table 2 shows the probability of propagating $P_a(u)$ and the probability to stop diffusion, $1 - P_a(u)$, in each category.

Given $P_a(u)$, a node u tosses a coin with the probability to have a head $P_a(u)$. If u gets a head, u keeps propagating, otherwise it stops propagation. We represent this random event as Y_u, which is defined formally as follows:

$$Y_u = \begin{cases} 1, & \text{decide to propagate information (head)} \\ 0, & \text{decide not to propagate information (tail)} \end{cases} \tag{22}$$

and

$$P_Y(y) = \begin{cases} P_p(u), & y \text{ is head} \\ 1 - P_p(u), & y \text{ is tail} \end{cases} \tag{23}$$

Now we discuss how $P_a(u)$ is used in conjunction with the closeness threshold θ_c and the probability of u influencing v, $w(u, v)$.

Without incorporating the categorization of user nodes, all users are assumed to engage in the influence diffusion. The probability of u activating one of its inactive neighbors, say v, is only dependent on $w(u, v)$ and θ_c. By introducing $P_a(u)$, we are differentiating nodes that are more actively engaged in influence diffusion from nodes that are less interested in propagating influence. Thus, we determine whether a node u will be propagating its

Table 2: Probability to Propagate or Stop

Category	$P_a(u)$	$1 - P_a(u)$
Innovator	0.90	0.10
Early Adopters	0.45	0.55
Early Majority	0.23	0.77
Late Majority	0.10	0.90
Laggards	0.05	0.95

influence to its inactive neighbor nodes in three steps. First, at the probability $P_a(u)$, node u will propagate its influence to its inactive neighbor nodes. Second, once u is in the state of propagating its influence, we use $w(u, v)$ and θ_c to determine probabilistically which of its inactive neighbor nodes will be activated to receive its influence. For those nodes that receive influence from u, say v, they will again use their respective $P_a(v)$ to probabilistically decide if v will continue to propagating us influence or stop propagation with 1-$P_a(v)$ probability.

Given a node u, we still need to determine which of the five categories to which u will belong. This will allow us to obtain $P_a(u)$, the group specific probability for propagating influence.

In order to categorize user nodes in a given social network, we propose to introduce the adoption probability group $A(u)$. A näive approach to compute $A(u)$ is to use the degree of its friends. We argue that using the degree of friends to compute $A(u)$ is not sufficient. As studied in [86, 23], some people want to have many friends just for increasing popularity, and others may agree friend requests in order not to be impolite. Thus the degree of friends may not accurately reflect the adopter probability and thus should be used as one of factors, rather than a sole factor, of activeness.

In our PSI model, we propose to combine node degree with the level of activities to compute $A(u)$. Activity is categorized into two groups: non-interactive activities (NA) and interactive activities (IA). According to [14], before buying a product, people tend to read reviews from those who already bought the product. If the reviewer provides more postings related to the product with detailed photos and descriptions, the followers would have

better understanding of the product. Such types of activities may generate different levels of influence on potential buyers' decision. We refer to these activities as non-interactive activities (NA). On the other hand, IA refers to the number of interactive activities between two users.

By combining both node degree and quantitative activities in both NA and IA categories, we formulate $A(u)$, the influence adoption probability group, as follows:

$$A(u) = \beta \cdot (\frac{NA(u) + IA(u)}{MAX(NA) + MAX(IA)}) + (1 - \beta) \cdot \frac{d_u}{MAX_{v \in V}(d_v)} \tag{24}$$

where $NA(u)$ is the number of non-interactive activities that u has performed, $IA(u)$ is the number of interactive activities that u did with her friends, d_u is the out-degree of u and $MAX_{v \in V}(d_v)$ is the maximum degree in the graph G and β is a balancing weight function, which carefully combines node degree and activities in computing $A(u)$. By varying β we can focus more on activities or degree of node u. When β is 0, $A(u)$ solely depends on the degree of node. On the other hand, if β is set to 1, then $A(u)$ is determined exclusively by the normalized number of interactive and non-interactive activities.

$A(u)$ represents the influence adoption probability of u and thus can be used to categorize node u into the appropriate influence adoption group. In this paper we use the five categories defined in [82] to classify social network nodes into five groups with respect to their willingness to adopt an innovation. They are Innovators, Early Adopters, Early Majority, Late Majority, and Laggards. In our first prototype, we set $P_a(u)$, node u's probability to propagate influence, as given in Table 2.

3.3.5 PSI with Rewards as Incentives

In this section we describe how to incorporate rewards as incentives to the probabilistic social influence (PSI) model in terms of reward effects and reward targets.

3.3.6 Reward Effects

In general, companies give out rewards to people in order to stimulate the product sales through "word of mouth" effect. Rewards can be given in many different forms and one way to model different forms of rewards is to define rewards in terms of benefits that a

user receives and the efforts that a user would need to make in order to receive rewards. We propose to formulate reward effects in terms of two factors: Efforts and Benefit. Some rewards require much more efforts in terms of time or monetary, whereas other rewards demand low efforts. For example, a credit card company C_1 offers 50,000 points after you spend \$3,000 in first 3 months. Another company C_2 offers \$10 credit back when you spend \$10 at a restaurant. C_1's promotion requires users to spend at least \$3,000 while C_2's promotion requires only \$10. Thus we use E to represent a scale of efforts, which ranges between 1 to 10. Similarly, we use B to denote a scale of benefit. C_1's promotion offers 50,000 points in their monetary system, which is worth to \$500, while C_2's promotion gives back \$10 credit. Two promotions also have different benefit scales, which is ranging between 1 to 10.

In PSI, we can formulate reward effects as the following formula:

$$R = c\frac{B}{E} \tag{25}$$

, where c is a normalized function that ensures R between 0 and 1.

Figure 18 shows the trend of reward effects by varying B and E at the same time. When we fix the effort scale and increase benefit scale from 1 to 10, then the reward effect also increase up to 10%. But when increase effort scale and the effect decreases as low as 0.1%

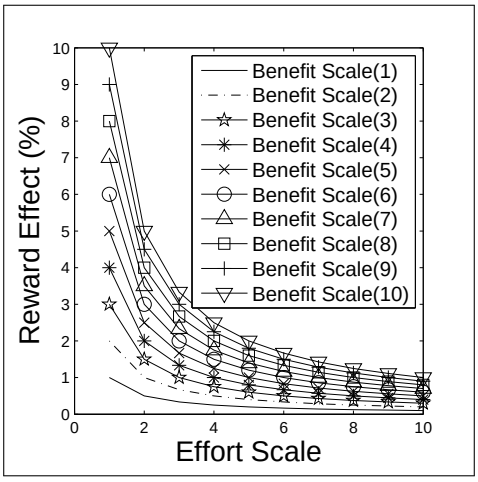


Figure 18: Reward Effect

Once a system parameter R is given, and u agrees to receive reward, there are two ways

to incorporate the reward incentive R into the PSI model. First, we incorporate the reward incentive into the probability to propagate influence, $P_a(u)$, by the following formula:

$$P_a(u) = P_a(u) + (1 - P_a(u))R \tag{26}$$

Alternatively we can also incorporate the reward incentive R into $A(u)$, the influence adoption probability group, to upgrade u to a higher adoption probability group.

3.3.7 Reward Target

For a social network of N nodes, the next question is then who should be given the reward under a limited budget constraint? This is very common when marketing companies have limited amount of budget to promote their products. They want to maximize the influence of the rewards so that the maximization of the utility of rewards can makes the largest possible number of people to buy the products. Given a limit of budget, randomly selection of marketing targets is inefficient. One way of distributing rewards is to select one group among the five groups: Innovator, Early Adopter, Early Majority, Late Majority, and Laggards.

Clearly, members in each group have similar adopter probabilities. Innovators have highest activation probability and laggards have the lowest. We will compare which group is the most effective group in terms of promoting new products. Each group of people is exposed to the reward but not all of people will get the reward because of the limited marketing budget. Each person in the group has a chance to take the reward or not. If the user takes the reward, then her probability to propagate, $P_a(u)$ will be increased by incorporating the reward effect into the Eq. (26).

3.4 Probabilistic InfluenceRank *Computation*

3.4.1 InfluenceRank Without Rewards

In Chapter 2, we defined activity based InfluenceCoverage as the number of nodes whose heat is greater than or equal to the system defined diffusion threshold θ_v at the end of the activity based iterative heat diffusion process. In the probabilistic based social influence diffusion model, v_i's InfluenceCoverage is the set of nodes that are activated

through chains of probabilistic activations, starting from the influence source v_i and ending when the probabilistic influence diffusion process converges.

Given InfluenceCoverage for all of vertices in the social network graph, we sort them descending order and give InfluenceRank to each vertex. In this section we describe a suite of the probabilistic InfluenceRank computation algorithms.

Let $G = (V, E)$ denote the target social network, $w(v_i, v_j)$ denote the probability of node v_i to activate node v_j for $v_i, v_j \in V, (v_i, v_j) \in E$ and θ_c denote the closeness threshold for activating influence diffusion. We can compute v_i's InfluenceRank in two steps:

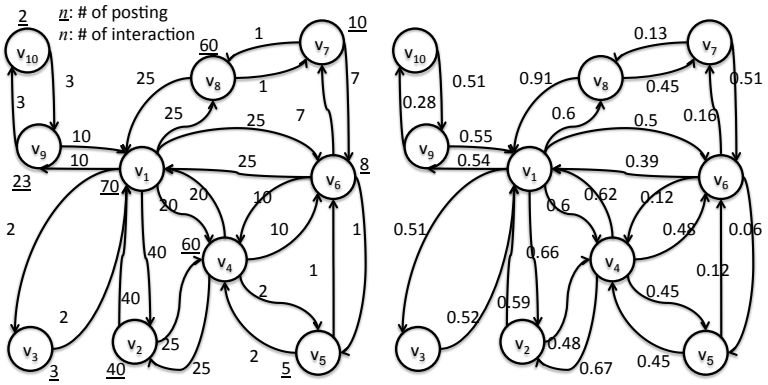
- In the first step, we select v_i as the influence diffusion source and v_i is active and all the other nodes in G are inactive. We then initiate the probabilistic influence diffusion process starting from v_i based on $w(v_i, v_j)$, θ_c, $A(v_j)$ for any $v_j \in V$ and $(v_i, v_j) \in E$.
- In the second step, we measure the number of nodes that are activated by v_i probabilistically by examining every outgoing link of v_i.

Figure 19(a) shows an example social network graph with 10 nodes and 26 edges. Each node has the number of postings in their profile page as shown in the number underlined. Each edge has a weight value defined by the number of interactions between two nodes. For example, v_9 has 23 postings and 6 interactions with v_{10} and 20 interactions with v_1. Recall that we use $\alpha \in [0, 1]$ as a weight function to balance between non-interactive activities and interactive activities in computing the edge weight probability $w(u, v)$. We set α to be 0.5. In this graph, $max(NA)$ is 70 from v_1. Thus $w(v_9, v_{10})$ is computed as follows:

$$\begin{aligned} w(v_9, v_{10}) =& \alpha \frac{NA(v_9)}{MAX(NA)} + (1-\alpha) \frac{IA(v_9, v_{10})}{\sum_{s:(v_9,s)\in E} IA(v_9, s)} \\ =& 0.5 \cdot \frac{23}{70} + (1 - 0.5) \cdot \frac{3}{3+10} \\ =& 0.28 \end{aligned}$$

The entire $w(u, v)$ is attached to each edges in the graph as shown in Figure 19(b).

By using the five categories: innovators (IN), early adopters (EA), early majority (EM), late majority (LM), and laggards (LA), we categorize the nodes in this example graph and



(a) Graph with Activities (b) Graph with Probability

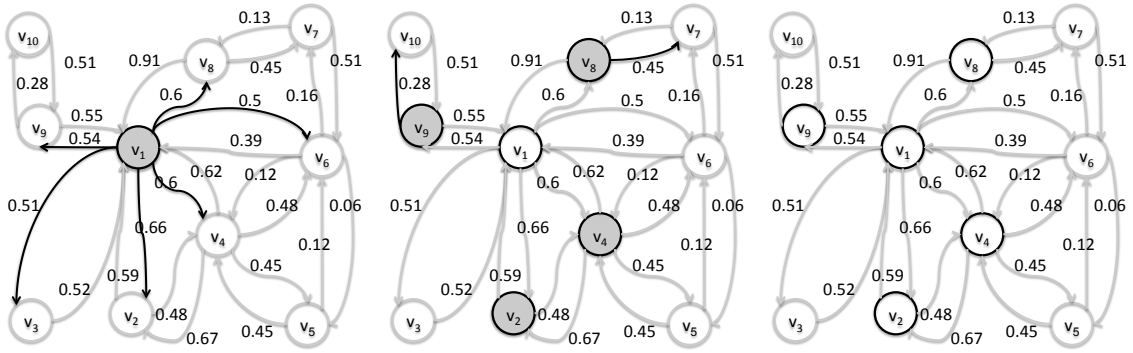
Figure 19: Probability $w(u, v)$

Table 3: $A(u)$ and Category of v_i

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
$A(u)$	1.00	0.47	0.10	0.60	0.18	0.47	0.21	0.34	0.24	0.10
$P(u)$	0.90	0.23	0.05	0.45	0.10	0.23	0.10	0.23	0.10	0.05
Group	IN	EM	LA	EA	LM	EM	LM	EM	LM	LA

compute their adopter probability category $A(u)$ using Eq. (24) as shown in Table 3. In this example, node v_1 is chosen as an innovator and v_3 and node v_{10} are considered as laggards since their adopter probability group $A(v_3)$ and $A(v_{10})$ are the lowest. Thus, if we set θ_c to 0.2, at $t = 0$ all nodes are inactive except the influence diffusion source v_1. Figure 20 shows each step of the diffusion activation process using the probabilistic influence diffusion approach. White circles with solid gray outline represent inactive nodes. Circles with black line are active nodes. Solid gray circles with solid black outline are the nodes that have been activated by v_1 and are now able to activate their inactive friends. According to Table 3, v_1 as an innovator node will propagate information to its friends with a probability $P_a(v_1)$, which is 90%, while v_3 and v_{10} will propagate information to their friends with probability$P_a(v_3)$ and $P_a(v_{10})$, which is 5%.

At $t = 1$, v_1 can activate her inactive neighbors with $P_a(v_1)$. This can be viewed as v_1 tossing a coin with a probability $P_a(v_1)$, which is 0.9, because v_1 is considered as an innovator as shown in Tables 2 and 3. We assume that Y_{v_1} returns 1, which means that all



(a) Activation at $t = 1$ (b) Activation at $t = 2$ (c) Activation at $t = 3$

Figure 20: Activation Steps

v_1's inactive friends can be activated by v_1 if $w(v_1, v_j) > \theta_c$ for $\forall j : (v_1, v_j) \in E)$ and v_j is inactive.

Let $\theta_c = 0.5$. Figure 19(b) shows that all of v_1's outgoing edges have the probability greater than θ_c and all of v_1's neighbors are inactive as shown in Figure 20(a). Thus, v_1 tries to activate all of its 6 neighbors, v_2, v_3, v_4, v_6, v_8, and v_9. By the probabilistic influence diffusion, we assume that v_1 activates all of its inactive nodes. In this example, four out of the six nodes, v_2, v_4, v_8, and v_9, are activated.

At $t = 2$, these newly activated nodes will follow their respective activation probability $P_a(v_i)$ for $i = 2, 4, 8, 9$ to continue propagating or to stop propagation. Let us assume that by tossing a coin Y_2, Y_8, Y_9 are 1 and Y_4 is 0. Thus, v_2, v_8, and v_9 have a chance to activate their inactive neighbors. Given that v_2 has no inactive friends to activate, v_2 terminates the diffusion process. Thus, only v_9 and v_8, marked by black solid edges, are considered at the step 3, each has only one inactive friend. v_8 tries to activate v_{10} and v_9 tries to activate v_7. However, given that $w(v_1, v_8) = 0.45$ and $w(v_1, v_9)$, we get $X_{v_1,v_8} = 0$ and $X_{v_1,v_9} = 0$. As a result, there are no newly activated nodes at $t = 2$.

At $t = 3$, the diffusion process converges since no new nodes are activated. Thus we stop the influence diffusion activation process for v_1 at $t = 3$ as shown in Figure 20(c). The InfluenceCoverage for v_1 is 4 since there are four nodes (v_2, v_4, v_8, v_9), which are included in the coverage of v_1's influence.

We have outlined the algorithm for computing InfluenceCoverage in a social network with an example. Given a node u, its InfluenceCoverage is computed by using u as the influence diffusion source and compute the coverage of nodes that can be activated by u using the PSI model.

3.4.2 Computing InfluenceCoverage by Matrix Multiplication

An alternative and more efficient approach to computing the InfluenceCoverage for all nodes in a social network graph $G = (V, E)$ is to use matrix multiplication.

Considering the running example given in Figure 19(a), we can formulate the computation of InfluenceCoverage for node v_1 as a matrix multiplication problem.

Let $S_t(i)$ denote the binary state of node v_i at time t and the value of $S_t(i)$ is either 1 (active) or 0 (inactive). For example, at $t = 0$, only v_1 is active. Thus we have $S_0(1) = 1$ and $S_0(i) = 0(1 < i < N)$ and N denotes the size of V.

v_1 makes a decision to keep propagate influence with the probability $P_a(1)$. Assume that by tossing a coin we get $Y_1 = 1$, which implies v_1 is propagating the influence to tis inactive neighbor nodes based on the threshold θ_c and the probability weight $w(v_1, v_j)$ on edge $(v_1, v_j) \in E)$ where v_j is inactive neighbor of v_1. This computation can be written as the multiplication of $S_0(1)$ and Y_1, namely $Y_1 \times S_0(1)$.

Assume that v_2 is activated by v_1 at $t = 1$. Then the state of v_2 at $t = 1$, $S_1(2)$, is 1. Thus, the activation results of v_2 at $t = 1$ can be represented by $S_1(2) = X_{1,2} \times Y_1 \times S_0(1)$.

In short, the activation process can be formulated as a three step process:

- Step 1: Define the state of the source of information diffusion by selecting a node v_1 at time $t = 0$, denoted as $S_0(1)$
- Step 2: Determine the probabilistically whether to keep propagating or not by multiplying Y_1 and $S_0(1)$, namely $Y_1 \times S_0(1)$
- Step 3: Compute the result of activation based on the probability $w(u, v)$, namely $X_{1,2} \times Y_1 \times S_0(1)$

In the case where both v_1 and v_4 are direct neighbors of v_2 and both nodes are active

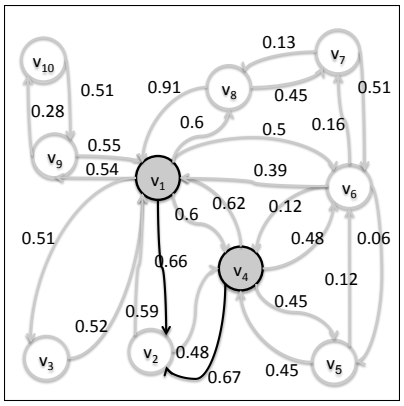


Figure 21: v_1 and v_4 try to activate v_2

at t, then both nodes will try to activate v_2 using PSI. The result of this activation can be written as $X_{1,2} + X_{4,2}$. If either nodes succeeds, then $X_{1,2} + X_{4,2} \geq 1$ holds, thus v_2 is activated; Otherwise v_2 remains as inactive. v_2's state at $t+1$ is expressed as follows:

$$S_{t+1}(2) = X_{1,2} \times Y_1 \times S_t(1) + X_{4,2} \times Y_4 \times S_t(4) \tag{27}$$

We can generalize the above equation to compute a node v_is state at time t+1 as follows:

$$S_{t+1}(i) = \sum_{(v_j, v_i) \in E} X_{j,i} Y_j S_t(j) \tag{28}$$

.

Note that for any node v_j that has no direct edge connecting to v_i at time t, we have $S_t(j) = 0$.

In order to compute InfluenceCoverage for all the vertices at the same time, we use the matrix representation. First, we use a state column vector S_t to represent the state of vertices at time t. For example, if both v_1 and v_4 are active and the rest of the nodes are inactive at $t = 0$, then S_0 is represented as follows:

$$S_0 = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{29}$$

Let $X_{u,v}$ denote the result of coin toss (activation). Node u has only one chance to activate its neighbor node v. Thus, $X_{u,v}$ can be pre-computed and stored as a $n \times n$ matrix

as follows:

$$X = \begin{bmatrix} X_{1,1} & X_{2,1} & X_{3,1} & \cdots & X_{n,1} \\ X_{1,2} & X_{2,2} & X_{3,2} & \cdots & X_{n,2} \\ & & \cdots & & \\ X_{1,n} & X_{2,n} & X_{3,n} & \cdots & X_{n,n} \end{bmatrix} \tag{30}$$

Similar to $X_{u,v}$, Y_u is also the result of coin toss to determine if node u is willing to activate its inactive neighbor nodes. Thus Y_u also can be pre-computed and stored as a $1 \times n$ matrix as follows:

$$Y = \begin{bmatrix} Y_1 & Y_2 & Y_3 & \cdots & Y_n \end{bmatrix} \tag{31}$$

By Eq. (29), (31), and (30), we can compute the state of vertices at time $t+1$ after activation at t as follows

$$S_{t+1} = XYS_t \tag{32}$$

Algorithm 4 provides a sketch of the computation of InfluenceCoverage for a given node v_i. We set the given node v_i as a influence source and active. Then perform a matrix multiplication until there are no more newly activated nodes. For each iteration we count the number of activated nodes for v_i's InfluenceCoverage.

Algorithm 4: InfluenceCoverage(v_i)

```
1  S_0 ← n × n zero matrix;
2  S_0(i) ← 1 ;                                  /* v_i becomes first active */
3  IC_i ← ∅ ;                                    /* initialize IC_i */
4  t = 0;
5  while Newly activated nodes exist do
6      S_{t+1} = XYS_t;
7      if ∀S_{t+1}(j) ≥ 1 then
8          S_{t+1}(j) ← 1;
9      end
10     IC_i ← IC_i ∪ {Newly activated nodes};
11     t ← t + 1;
12 end
```

3.5 InfluenceRank *Computation Algorithms*

We have explained probability-based influence model and an algorithm for computing InfluenceCoverage. In this section we present how to assign InfluenceRank and select marketing target nodes such that the effect of marketing is maximized using InfluenceRank. In general, we can describe the problem as selecting a subset of nodes in a social network, which can provide the maximum coverage of social influence. This problem is NP hard and given that the maximum coverage function has two nice properties: monotonic and diminishing return. We can use a greedy algorithm to approximate the theoretical optimal solution with a strong (66.6%) theoretical guarantee.

Our social influence model also uses the three algorithms to assign InfluenceRank. (a) independent InfluenceCoverage, (b) locally minimal overlap InfluenceCoverage, and (c) globally minimal overlap InfluenceCoverage.

For each three criteria, we use the same structure of algorithm.

- Step 1: Compute X. Result of activation is the same as coin toss. Once a probability is given, the result can be pre-computed and stored as a matrix X.
- Step 2: Compute IC_i. For each node v_i we compute corresponding IC_i. IC_i can be computed individually or together with other active nodes.
- Step 3: Sort IS_i and select top-k nodes order by IC_i.

Independent InfluenceCoverage

Given a social network composed of n nodes, Algorithm 5 shows the steps how to rank InfluenceRank using independent InfluenceCoverage. The basic steps of this algorithms is first to calculate InfluenceCoverage of each node v_i and then sort nodes descending order of InfluenceCoverage size.

This algorithm is the simplest one in terms of implementation. However, if we select top-k users using this InfluenceRank, it ignores the potential overlaps of top-k influential sets. Some of nodes that are selected as k most influential nodes may have large overlapping activated nodes. Table 4 shows InfluenceCoverage for 5 nodes: v_1, v_2, v_3, v_4, v_5. Based

Algorithm 5: Independent InfluenceCoverage

```
Compute Y;
Compute X;
foreach v_i ∈ V do
    IC_i ← InfluenceRank(v_i) ;                    /* initialize IC_i */
end
Sort {IC_i | v_i ∈ V} by set size;
Rank V by the sorted order;
```

Table 4: InfluenceCoverage

Node	InfluenceCoverage	Locally Minimal Overlap InfluenceCoverage
v_1	$v_{11}, v_{12}, v_{13}, v_{14}$	∅
v_2	v_{11}, v_{12}, v_{13}	∅
v_3	v_{11}, v_{12}	∅
v_4	v_{15}, v_{16}	v_{15}, v_{16}
v_5	v_{16}	v_{16}

on the Algorithm 5, top-2 influential nodes are v_1, v_2. But v_2's coverage is completely overlapping with v_1's. Therefore, we need to consider InfluenceCoverage overlap when assigning InfluenceRank.

Locally Minimal Overlap InfluenceCoverage

This algorithm is a localized greedy algorithm. For each node v, the Algorithm first computes IC_v, an InfluenceCoverage. At first, a node v whose IC_v is the largest is selected as the first seed of top-k influential nodes. For each remaining node u, we compute the locally minimal overlap InfluenceCoverage, LC_u, which is a difference between two coverages, a universal coverage, IC, from a set of previously added highest InfluenceRank nodes and a coverage from one of the remaining nodes for every node. We select a node u as a next highest InfluenceRank node whose LC_u is the largest. After adding a node u as the next highest InfluenceRank node, we remove u from G and set it G'. This process iterates until G and G' are different. For example, v_1 is selected as the highest InfluenceRank node. The universal coverage IC is now IC_{v_1}. Locally minimal overlap InfluenceCoverage , LC_u, is shown in Table 4. Other remaining nodes, v_2, v_3, v_4, computes LC_u by computing difference from IC, which is IC_{v1}. v_2's coverage is completely

overlapping with v_1's. Therefore, the overlap for v_2 is empty. v_3 also has the empty set due to the same reason. v_4 and v_5 has non empty minimal overlap InfluenceCoverage. But v_4 has the larger set and selected as the next highest InfluenceRank node.

Algorithm 6: Locally Minimal Overlap InfluenceCoverage

```
1  Line 1 to 4 in Algorithm 1;            /* Independent InfluenceCoverage */
2  IC ← ∅;
3  G' ← G;
4  V_local ← 0;
5  while G' ≠ G do
6      foreach v_i ∈ (G − V_local) do
7          LC_i ← IC_i − IC;
8      end
9      Select v_i that has max(LC_i);
10     V_local ← V_local ∪ {v_i};
11     G' ← G − v_i;
12     IC ← IC ∪ IC_i;
13 end
14 Sort v ∈ V_local by the order of addition to V_local;
15 Rank v ∈ V_local by the sorted order;
```

Globally Minimal Overlap InfluenceCoverage

Although Algorithm 6 generates a good InfluenceRank, it is a locally greedy algorithm since it does not consider cases where multiple sources of diffusion exist. For example, on a launch of a new iPhone, multiple bloggers post their own reviews. The information that one person receives may come from several diffusion sources. People may read multiple reviews before deciding to purchase the iPhone. Thus, we need to devise a global greedy the algorithm to model the multiple sources of information diffusion.

First part of Algorithm 7 is similar to Algorithm 6. It computes individual InfluenceCoverage then selects a node v whose InfluenceCoverage is the largest (Line 3). Then we set v as an initial seed of source of information diffusion (Lines 4-6). We denote IC as the universal InfluenceCoverage for the set of influential nodes V_{global}. Now we use Hill-climbing algorithm to simulate multiple source of influence diffusion. First, for each node v_i not in V_{global}, we add temporarily v_i to V_{global}, then compute InfluenceCoverage, IC_i, which simulates multiple sources of information diffusion (Lines 12-16). Given two coverages, IC_i and IC, we define globally minimal overlap InfluenceCoverage, GC_i, as a difference

Algorithm 7: Globally Minimal Overlap InfluenceCoverage

Line 1 to 4 in Algorithm 1; `/* single-source information diffusion */`
$IC \leftarrow \emptyset$;
Find v_i that has $\max(IC_i)$;
$IC \leftarrow IC_i$;
$V_{global} \leftarrow v_i$;
$G' \leftarrow G$;
while $G' \neq G$ **do**
    $S_0 \leftarrow n \times n$ zero matrix;
    foreach $v_i \in V_{global}$ **do**
        $S_0(i) \leftarrow 1$; `/* multiple starting points */`
    end
    foreach $v_i \in (V - V_{global})$ **do**
        $S_t(i) \leftarrow 1$;
        $IC_i = \text{InfluenceCoverage}(V_{global} \cup v_i)$; `/* multi-source diffusion */`
        $GC_i \leftarrow IC_i - IC$;
    end
    Find v_i that has $\max(GC_i)$;
    $V_{global} \leftarrow V_{global} \cup v_i$;
    $G' \leftarrow G - v_i$;
end
Sort $v \in V_{global}$ by the order of addition to V_{global};
Rank $v \in V_{global}$ by the sorted order;

between IC_i and IC. After computing GC_i for all remaining nodes, we select v_i that has the largest GC_i as the next highest InfluenceRank node.

3.6 Experiments

In this section we report our experimental evaluation of the performance and effectiveness of our PSI model. Our experiments are conducted with two objectives. First, we want to show the effects of parameters that we present in our PSI models such as α, a weight function for balancing between NA and IA, θ_c, closeness threshold, and β, a balancing weight function between the sum of NA and IA and degree. Each parameter affects the number of activated nodes. These parameters should be carefully chosen for different types of SNS. Our experiments will be a good guidance in selecting those parameters. Second, we want to evaluate the performance of our PSI models with or without rewards against topology-based and näive activity-based approach. We show that probability and incentive approach has up to 7 times bigger activation coverage than previous approaches.

3.6.1 Datasets

Three datasets are used in this experiments such as DBLP[10], Epinions[1], and and Facebook[2]. DBLP datasets consist of paper authors as nodes and their co-authorship as edges. For example, two authors u and v wrote two papers, and u and w wrote three papers, then u and v are connected by $E(u, v)$ and u and w by $E(u, w)$. Because u wrote two papers with v and three papers with w, we set $IA(u, v)$ as 2 and $IA(u, w)$ as 3. Note that u wrote total 5 papers with u and v. Writing five papers can be considered not only as interactive activities but also non-interactive activities. Therefore, we set $NA(u)$ as 5. DBLP dataset has 4,768 nodes and 32,020 edges.

Facebook is a social network service that provides a profile page for each user. Users can update their profile page, post photos, and leave comments on her posting or friends' postings. When a user u posts something on her page, we consider it as a non-interactive activities. If u leaves comments on her friend's posting, it is considered as an interactive activity. By counting NA and IA, we compute $NA(u)$ and $IA(u, v)$. We launched a Facebook app and in total 273 users used the app. Once a user u allows us to use the

private information, we extract their friends relationship information. For example, one user may have 400 friends. Then from one user we can create 401 user nodes. Some users have more than 1,000 friends. By doing this, we can create 76,954 user nodes and 1,121,861 friendship relationships from 273 users.

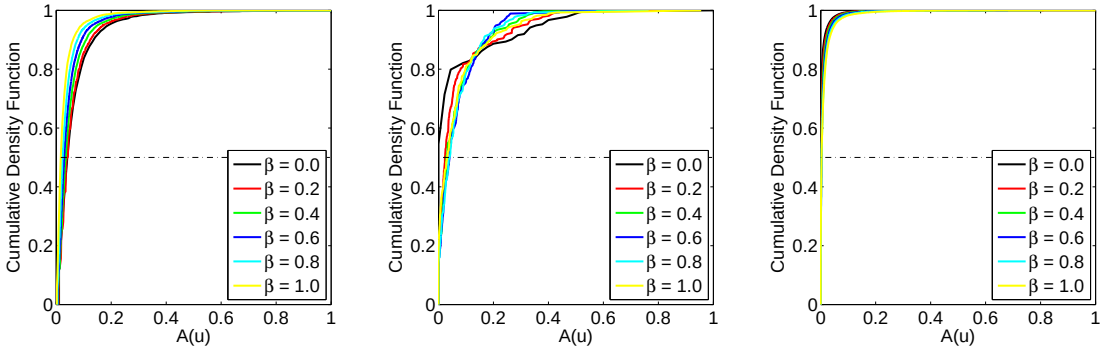
Massa[70] collected data from Epinions, a website for consumer reviews and trust networks. Epinions provide a system that users who bought products can leave reviews on them. Then potential buyers read reviews and determine if they buy the product or not. The potential buyers do not solely rely on the reviews but reviews have influence on users. Therefore Epinions is a good dataset to gauge influence. On Epinions, users post reviews. We consider this reviews as NA. For IA, we use a trust list. For example, if users u and v posted some reviews. When a user w likes u's reviews and does not v's reviews, then w creates a trust list by adding u and does a block list by adding v. Next time when w visits Epinions, reviews from w's trust list will be shown and one from w's block list will be filtered out. We create $E(u, v)$ from the trust link. Epinions dataset 49,288 nodes and 487,002 edges.

3.6.2 Effects of β

The first set of experiments focus on β, which is used to compute $A(u)$ and plays a role of balancing a weight between the number of activities and the degree of u. Figure 22 shows the adopter probability category $A(u)$ by varying β. x-axis is the value of $A(u)$ and y-axis is the cumulative density function. We vary the balancing weight β from 0 to 1.

DBLP data and Facebook data show that when we increase β, $A(u)$ decreases. Greater value of β means we consider activities are more important factor in computing $A(u)$ than degree of a node. Activity information consists of two values; NA and IA. The sum of two values are normalized divided by the sum of two values $MAX(IA) + MAX(NA)$. In order to get high $A(u)$, both NA and IA should be also large. But not all nodes have two large values. Therefore when we increase β, $A(u)$ is decreasing. On the other hand, Epinions dataset has different property. If we increase β, $A(u)$ also increases. Epinions dataset has NA which is the number of reviews and does not have IA. Thus, when we increase and set

β to be 1, the value of $A(u)$ is computed by $NA(u)$ only. However, if we set low β such as 0, then $A(u)$ completely depends on d_u. Compared to $NA(u)$, d_u is lower and the standard deviation is high such as mean 9 and std 32, respectively. Big difference in d_u means lower value of $\frac{d_u}{MAX_{u \in V}(d)}$. From now on, we set β as 0.5 to weight on evenly both the number of activities and the degree of u.



(a) $A(u)$ (DBLP) (b) $A(u)$ (Facebook) (c) $A(u)$ (Epinions)

Figure 22: Adopter Probability Category $A(u)$

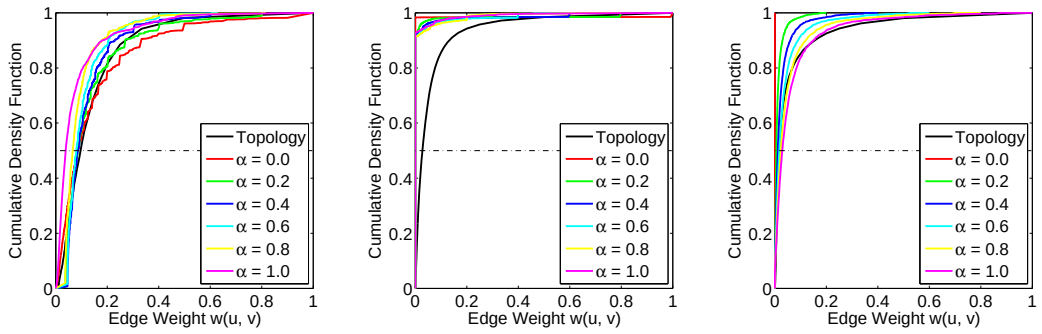
3.6.3 Effects of α

The parameter α is used in computing $w(u, v)$, the probability for u to activate v. α is a balancing weight between NA and IA in computing $w(u, v)$. However, Epinions dataset has only NA values, thus we designed an experiment for DBLP and Facebook datasets only. In this experiments, we set θ_c as 0.05 uniformly for all of nodes and β as 0.5 to evenly weight on activities and the degree of node. Due to the value of θ_c, nodes with $w(u, v)$ less than 0.05 are excluded for activating process.

Figure 23 shows the cumulative density function of $w(u, v)$ for each dataset. When we increase α, a weight parameter balancing between NA and IA, $w(u, v)$ decreases. In computing $w(u, v)$ we normalize $NA(u)$ by dividing from $MAX(NA)$ and $IA(u, v)$ by dividing from $\sum IA(u)$. In other words, $NA(u)$ is normalized over the entire NA values while $IA(u, v)$ is normalized among u's IA values. Then the difference between $NA(u)$ and $MAX(NA)$ might be larger than the difference between $IA(u, v)$ and $\Sigma IA(u)$. The bigger difference, the smaller fraction value, which means $\frac{NA(u)}{MAX(NA)}$ might be smaller

than $\frac{IS(u,v)}{\Sigma IA(u)}$. Thus, when we increase α, $w(u,v)$ is also decreasing as shown in Figure 23(a). Note that Epinions dataset has no IA information and $w(u,v)$ completely weight on NA. Thus, when we set α to be 0, then $w(u,v)$ is also 0. As we increase α, $w(u,v)$ also increases because $w(u,v)$ weights more on $\frac{NA(u)}{MAX(NA)}$ as shown in Figure 23(c).

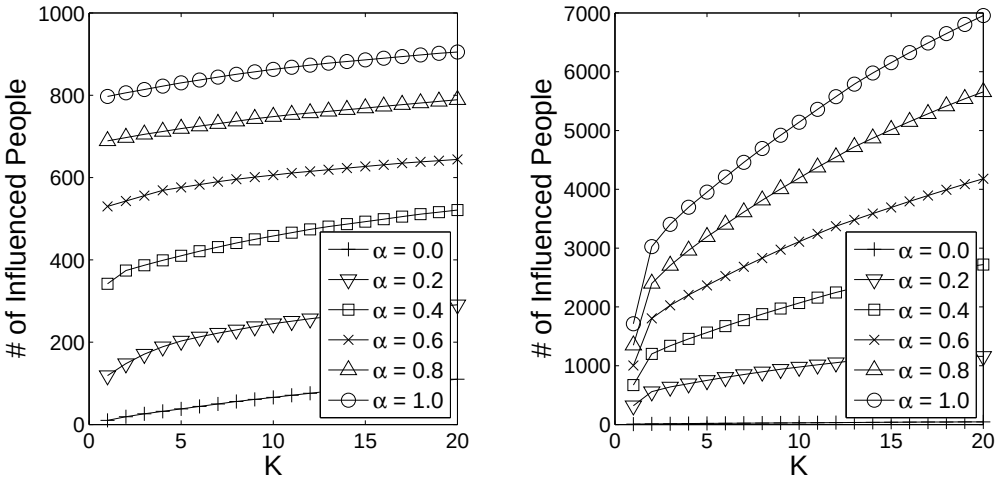
Note that Figure 23(b) shows that more than 90% of nodes have $w(u,v) < 0.01$. For collecting Facebook dataset, we request 273 users to take a part in our experiments. Thus, for each node u in these 273 nodes, we can get $NA(u)$, $IA(u)$, and d_u and we extract u's friend network which results in 76,954 friends and 1,121,861 friendship. For each node v in 76,954 extracted nodes, we have no $NA(v)$ and very limited information of $IA(v)$ and d_v. For example u is one of 273 Facebook app users, v is not a Facebook app user. If v leaves a comment on u's photo, we create two nodes u and v, edges $E(u,v)$ and $E(v,u)$, and set $IA(v,u)$ as 1 and $NA(v)$ as 0. Due to the way of constructing the social network graph, some nodes have NA, while other do not have NA. Also some nodes have high IA, while others have lower IA. This distribution results in lower $w(u,v)$ for node u that is not one of 273 Facebook App users. Therefore, more than 90% of $w(u,v)$ are lower than 0.01.



(a) $w(u,v)$ (DBLP) (b) $w(u,v)$ (Facebook) (c) $w(u,v)$ (Epinions)

Figure 23: Probability $w(u,v)$

Figure 24 shows the effect of α. x-axis shows the number of top-k users and y-axis shows the number of activated nodes. In both datasets, when we set α to be higher, the number of influenced people tends to be also higher. In order to have a fair comparison we will set α as 0.6 in the next experiments.

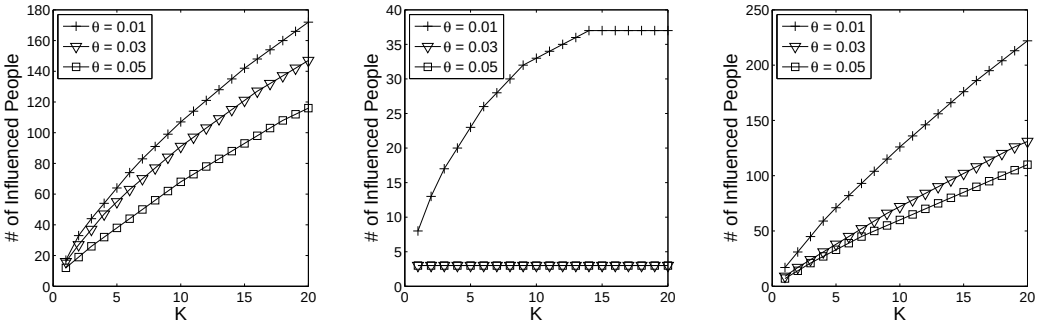


(a) DBLP (b) Facebook

Figure 24: Effects of α

3.6.4 Effects of θ_c

The parameter θ_c is a value used for filtering out acquaintances. If $w(u, v)$ is less than θ_c, we consider u and v is not close enough for activation process. Figure 25 shows the effect of θ_c for all three datasets. We vary θ_c from 0.01 to 0.05. If we increase θ_c then more people are excluded for activating process because there will be more edges with $w(u, v) < \theta_c$. Once the number of nodes to be target of activation is decreased, the number of activated nodes also decreases. In the following experiments we set θ_c as 0.05.



(a) DBLP (b) Facebook (c) Epinion

Figure 25: Effects of θ_c

Note that in Facebook dataset the lines for $\theta_c = 0.03$ and $\theta_c = 0.05$ are the same. We

explained why more than 90% of nodes in Facebook data have lower $w(u, v)$ in section 3.6.3. 95% of nodes in Facebook dataset has $w(u, v)$ less than 0.01. Thus when we set θ_c to be larger than 0.01, most of nodes do not take a part in information diffusion and the number of activated nodes are extremely low. Note that when we increase θ_c larger than 0.01, the number of nodes with $w(u, v)$ larger than θ_c is the same. Therefore, the number of activated nodes for $\theta_c = 0.03$ and $\theta_c = 0.05$ are the same. From now on we set θ_c as 0.05 except for Facebook dataset (θ_c as 0.01).

3.6.5 Effects of Reward Effect, R

Figure 27 shows how the reward effect R affects the number of activated nodes. In this experiment we set α as 0.6, β as 0.5, and θ_c as 0.05. For each dataset we vary R from 0.01 to 0.2. If we set R to be 0.01 than it boosts the $P_a(u)$ 1%. If we set R to be 0.2, then $P_a(u)$ is increased 20%. We also varied the marketing target. For each experiment we select only one group as a marketing candidate. Individuals in the candidate group have chance to get the reward. If the individual accepts the reward, then $P_a(u)$ is boosted by R we set. Therefore the impact of rewards is to decrease the probability of u to become a stopper.

x-axis shows the number of top-k users and y-axis shows the number of activated nodes. For each dataset, we performed the five experiments. For each experiment, one group is selected as a marketing candidate. We give individuals in the selected group a chance to take reward. For example, Innovator line in the chart shows the total number of influenced nodes when we assign incentives to only individuals in innovator group.

In DBLP datasets, when we target innovators, the number of activated nodes are highest. Early adopters may be the alternative target for marketing but the effect of rewards for the early adopters is only slightly bigger than other groups. Remaining three groups, early majority, late majority, and laggards, are not appropriate targets for reward. Although R increases the probability to be active, $P_a(u)$, and u becomes active, u may still have a very low $w(u, v)$. Innovators and Early Adopters usually have higher $w(u, v)$ because they have large $NA(u)$ and $IA(u)$. However, Early Majority, Late Majority and Laggards may have lower $w(u, v)$ due to their low activities. Therefore, regards less of R's boosting in $P_a(u)$,

low values in $w(u, v)$ result in the low number of activated nodes.

In Epinions dataset, we do not have IA information. Therefore $w(u, v)$ values are also low. Even Innovators may have lower $w(u, v)$. Due to the lower $w(u, v)$, R is not effective for all types of nodes when R is lower than 0.1. Like DBLP dataset, individuals in the innovator group is the most effective marketing target for rewards.

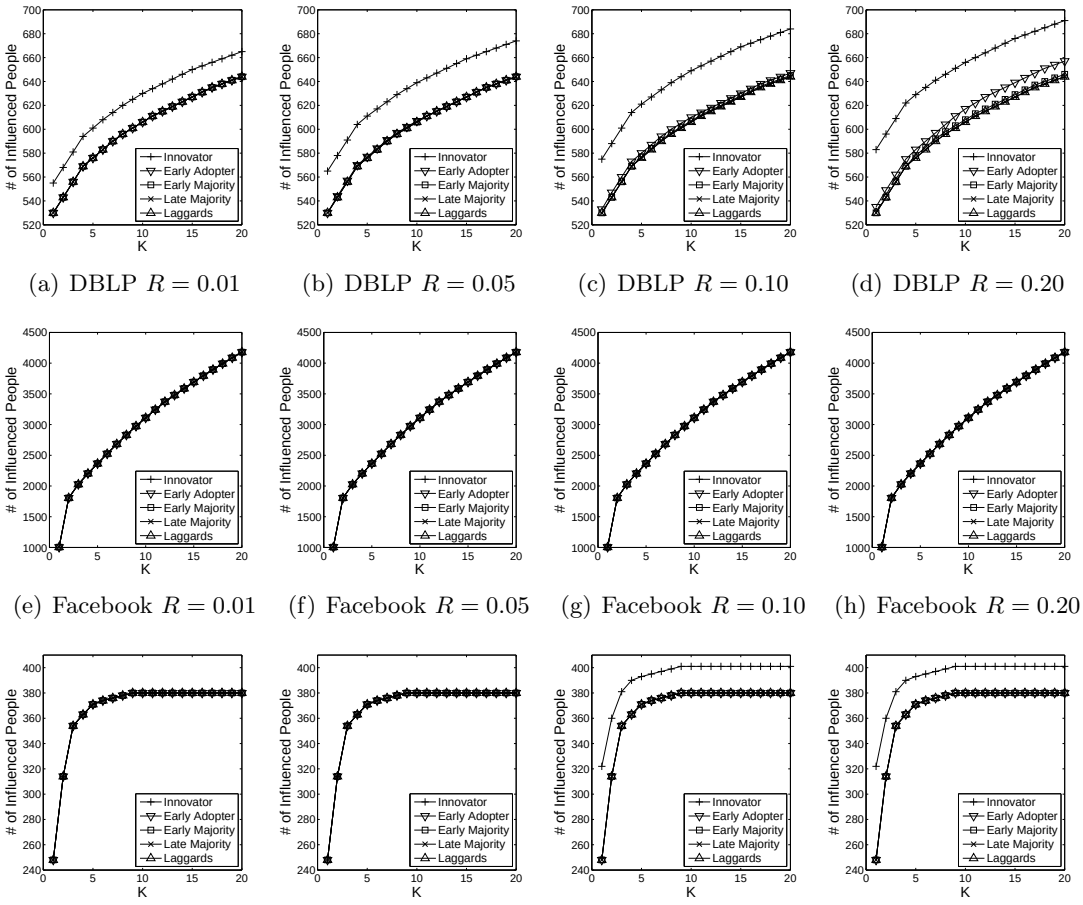
For Facebook datasets, the number of activated nodes are the same for all five marketing target groups as shown in Figures 26(e), 26(f), 26(g), 26(h). More than 90% of $w(u, v)$ values in Facebook datasets are lower than θ_c. This is due to the way of constructing social graph. $w(u, v)$ is computed using both $NA(u)$ and $IA(u)$ but Only 273 users have NA and some users have IA. Thus, $w(u, v)$ is extremely low. Although rewards boosts the probability to participation, extremely low $w(u, v)$ diminishes rewards effects as shown in Figures 26(e), 26(f), 26(g), 26(h). These figures may mislead that rewards are not effective at all. For Facebook dataset, we modified reward effect so that R can boost both $P_a(u)$ and $w(u, v)$. Boosting $w(u, v)$ is computed as follows:

$$w(u, v) = w(u, v) + (1 - w(u, v))R \tag{33}$$

A node u who agrees to get a reward will have a boosted $P_a(u)$, which makes u more actively participate in propagating information diffusion, and increased $w(u, v)$, which allows u to have higher chance to succeed in activating a neighbor node v. We applied this modified Eq. 33 and did the same experiments over Facebook dataset. Figures 27(a), 27(b), 27(c), 27(d) show the result of the experiment. Similar to other datasets, innovators respond more actively over rewards.

3.6.6 Comparions

Lastly, we conducted an experiment to show the performance of PSI model with and without rewards against topology-based approach. We set α as 0.6, β as 0.5, and θ_c as 0.05. For Facebook datasets, we use modified Eq. 33 so that we make R effective. x-axis is the number of top-k influential users and y-axis is the number of activated nodes. In all three datasets, Topology-based approach has the lowest number of influenced people. As explained in the previous sections, Topology-based approach sets the uniform probability to activate friends.



(a) DBLP $R = 0.01$ (b) DBLP $R = 0.05$ (c) DBLP $R = 0.10$ (d) DBLP $R = 0.20$

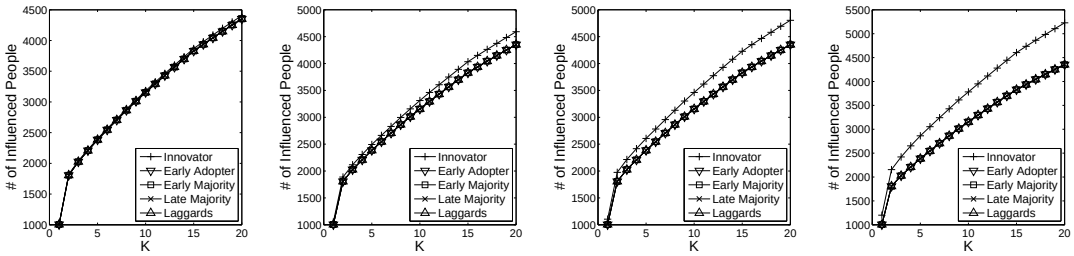
(e) Facebook $R = 0.01$ (f) Facebook $R = 0.05$ (g) Facebook $R = 0.10$ (h) Facebook $R = 0.20$

(i) Epinions $R = 0.01$ (j) Epinions $R = 0.05$ (k) Epinions $R = 0.10$ (l) Epinions $R = 0.20$

Figure 26: Effects of R

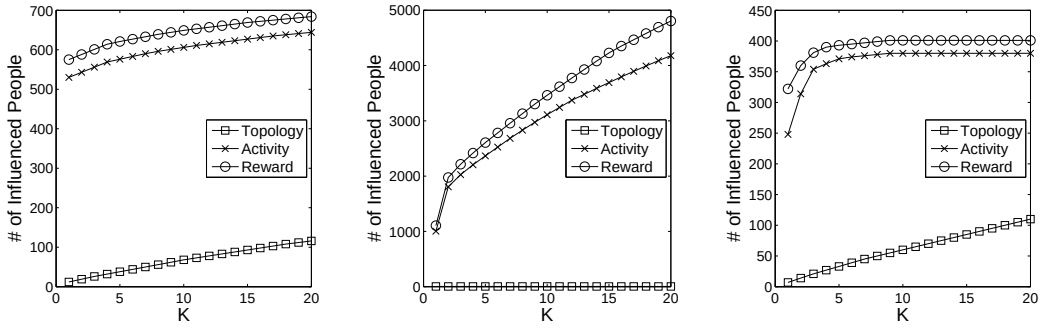
When a degree is high, then all of friends may not be activated because $\frac{1}{d_u}$ can be very low. But when we consider activity information, we differentiate $w(u, v)$ so that some of close friends are activated and this activation continues to friends of friends.

On top of probability and activity-based approach, we select Innovators as marketing target and give them a chance to accept rewards. Figure 28 shows that our PSI model with and without rewards have the larger number of influenced nodes, especially for DBLP dataset, the number of influenced nodes by PSI with rewards is 7 times more than the number of influenced nodes by topology-based approach.



(a) Facebook $R = 0.01$ (b) Facebook $R = 0.05$ (c) Facebook $R = 0.10$ (d) Facebook $R = 0.20$

Figure 27: Effects of modified R for Facebook



(a) DBLP (b) Facebook (c) Epinion

Figure 28: Comparisons of Three Approaches

3.7 Related Work

In a viral marketing campaign, marketing companies send a marketing message and encourage customers to forward this message to their friends. It is an important issue to find the top k influential people in SNS in the context of viral marketing so that the selected people maximize the spread of influence under certain influence propagating models. Researchers have studied how to find those people in SNS [71, 60, 52, 27, 15, 91, 25]. We also provide activity-based social influence model and algorithms in Chapter 2. A SNS is the network of relationships and interactions among members. Therefore instead of taking topology of SNS itself, we adopt other attributes in SNS in order to get more accurate social influence.

Although the activity-based approach performs better than topological approaches, it still has issues. Because it is the extended version of topological heat diffusion, fundamental problems are not solved. We explain in detail those issues.

3.8 Conclusion

We have presented the probabilistic social influence model (PSI) based on enhanced activity information. Compared to previous heat diffusion and topology-based probability models, our model contributes in three aspects. First, in order to express the real world more accurately, we introduce various system parameters. These parameters such as an weight function for balancing between NA and IA for computing $w(u, v)$, a closeness threshold θ_c to filter out acquaintances, and β for balancing between the number of activities and the degree of node u. Second, we develop a multi-scale incentive model so that any types of incentives can be used once it is normalized as a number between 0 and 1. By this incentive model, we show the boosting effect that the real incentives actually have. Reward effects boost $P_a(u)$ so that a node u may have lower chance to be a stopper and higher chance to activate her friends. Finally we conduct an extensive series of experiments on various parameters and performance of our models. From the experimental results, we show that by tuning the parameter we can precisely model the real world. Our PSI model with rewards maximizes propagating innovations which can be a new idea or new product over the given network.

CHAPTER IV

INFORMATION DISSEMINATION WITH SPATIAL ALARMS

4.1 Introduction

Time-based alarms are effective reminders of future events that have a definite time of occurrence associated with them. Just as time-based alarms are set to remind us of the arrival of a future reference time point, spatial alarms are set to remind us of the arrival of a spatial location of interest. Thus, spatial alarms can be modeled as location-based triggers which are red whenever a mobile user enters the spatial region of the alarms. Spatial alarms provide critical capabilities for many location-based applications ranging from real time personal assistants, inventory tracking, to industrial safety warning systems.

A mobile user can define and install many spatial alarms; each alarm is typically shared by one or many other users. Alarms can be classified into three categories based on the publish-subscribe scope of the alarm as private, shared or public alarms. Private alarms are installed and subscribed to exclusively by the alarm owner. Shared alarms are installed by the alarm owner with a list of $k(k > 1)$ authorized subscribers and the alarm owner is typically one of the subscribers. Mobile users may subscribe to public alarms by topic categories or keywords, such as "traffic information on highway 85North", "Top ranked local restaurants", to name a few. Each alarm is associated with an alarm target which specifies the location of interest to the user; a region surrounding the alarm target is dened as the spatial alarm region. The alarm trigger condition requires that subscribers of the alarm be notified as soon as they enter the spatial alarm region.

Processing of spatial alarms requires meeting two demanding objectives: high accuracy, which ensures no alarms are missed, and high scalability, which guarantees that alarm processing is efficient and scales to large number of spatial alarms and growing base of mobile users. The conventional approach to similar problems involves periodic evaluations at a high frequency. Each spatial alarm evaluation can be conducted by testing whether the

user is entering the spatial region of the alarm. Though periodic evaluation is simple, it can be extremely inefficient due to frequent alarm evaluation and the high rate of irrelevant evaluations. This is especially true when the mobile user is traveling in a location that is distant from all her location triggers, or when all her alarms are set on spatial regions that are far apart from one another. Further, even a very high frequency of alarm evaluations may not guarantee that all alarms will be successfully triggered. The spatial continuous query approach would process a spatial alarm by transforming the alarm into a user-centric continuous spatial query. Given the alarm region of radius r around the alarm target and the mobile users current location, the transformed spatial query is dened by the query range r with the mobile alarm subscriber as the focal object of the query. The query processor checks if the obtained query results contain the alarm target object. This process repeats periodically until the alarm target is included in the query results at some future time instant. The obvious drawback of this approach is the amount of unnecessary processing performed in terms of both the number of evaluations and the irrelevant query result computation at each evaluation. A more detailed discussion of the weaknesses can be found in our technical report [17].

Spatial alarms can be processed using server-based infrastructure or client-based architecture. A server-based approach must allow optimizations for processing spatial alarms installed by multiple mobile clients, whereas a client-based approach focuses more on energy-efficient solutions for evaluating a set of spatial alarms installed on a single client. Bearing in mind the problems inherent with the continuous spatial query evaluation approach and drawbacks of the periodic alarm evaluation approach, we develop a safe period-based alarm evaluation approach. The goal of applying safe period optimization is to minimize the amount of unnecessary alarm evaluations while ensuring zero or very low alarm miss rate. The other technical challenge behind safe period optimization is to minimize the amount of safe period computation, further improving system scalability and achieving higher throughput. We describe our basic approach for safe period computation in the next section and address the challenge of reducing the amount of safe period computations in Section 3. We evaluate the scalability and accuracy of our approach using a road network simulator

and show that our proposed framework offers significant performance enhancements for the alarm processing server while maintaining high accuracy of spatial alarms.

4.2 Safe Period Computation

Safe period is dened as the duration of time for which the probability of an alarm being triggered for a subscriber is zero. Consider a subscriber $S_i(1 \leq i \leq N)$ and a spatial alarm $A_j(1 \leq j \leq M)$, where N is the total number of mobile users and M is the total number of alarms installed in the system. The safe period of alarm A_j with respect to subscriber S_i, denoted by $sp(S_i, A_j)$ can be computed based on the distance between the current position of Si and the alarm region R_j , taking into account the motion characteristics of Si and alarm target of A_j . Concretely, for alarms with mobile subscribers and static targets, the two factors that influence the computation of safe period $sp(S_i, A_j)$ are (i) the velocity-based motion characteristic of the subscriber Si, denoted by $f(V_{S_i})$ and (ii) the distance from the current position of subscriber Si to the spatial region Rj of alarm Aj , denoted by $d(S_i, R_j)$. Thus the safe period $sp(S_i, A_j)$ can be computed as follows:

$$sp(S_i, A_j) = \frac{d(S_i, R_j)}{f(V_{S_i})}(1) \tag{34}$$

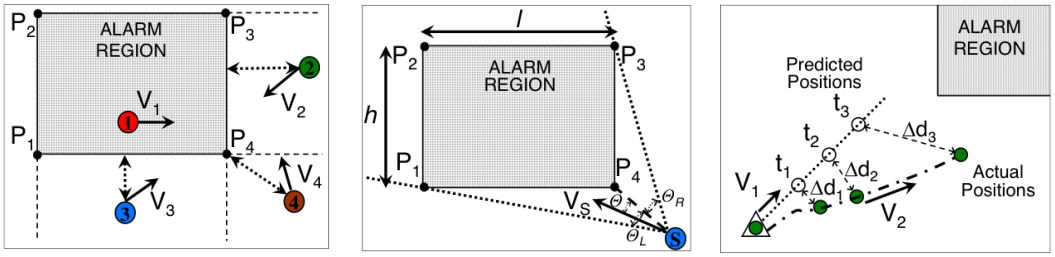
4.2.1 Distance Measurements

We use Euclidean distance as the basic distance measure for safe period computation. It measures the minimum distance from the current position of the mobile user, denoted as $P_m = (x_m, y_m)$, to the spatial alarm region R. Consider a spatial alarm region R covering the rectangular region represented by four vertices of a rectangle (P_1, P_2, P_3, P_4), as shown in Figure []. The minimum Euclidean distance from P_m to the spatial alarm region R, denoted by $d_{m,R}$, can be computed by considering the following four scenarios: (1) when the mobile subscriber lies inside the spatial alarm region the distance $d_{m,R}$ is zero; (2) when the mobile subscriber is within the y scope of the spatial alarm region, the minimum euclidean distance is the distance from the mobile subscriber to the nearer of the two spatial alarm edges parallel to the x-axis; (3) when the mobile subscriber is within the x scope of the spatial alarm region, minimum euclidean distance is the distance from the mobile subscriber

to the nearer of the two spatial alarm edges parallel to the y-axis; and (4) when the mobile subscriber is outside both the x and y scope then the distance is the minimum of the euclidean distance to the four vertexes. Formally, $d_{m,R}$, the minimum Euclidean distance from mobile position P_m to the spatial alarm region R, is computed using the following formula:

$$d_{m,R} = \begin{cases} 0 & , x_1 \leq x_m \leq x2, y_1 \leq y_m \leq y_2 \\ min(|x_m - x_1|, |x_m - x_2|) & , y1 \leq y_m \leq y_2 \\ min(|y_m - y_1|, |y_m - y_2|) & , x1 \leq x_m \leq x_2 \\ min(d_{m,1}, d_{m,2}, d_{m,3}, d_{m,4}) & , otherwise \end{cases}$$

, where $d_{m,k}, k \in \{1, 2, 3, 4\}$ denotes the Euclidean distance from P_m to rectangle vertex P_k. The distance function $d_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$ is used to compute the Euclidean distance between two points P_i and P_j.



(a) Euclidean Distance (b) Steady Motion (c) Dead Reckoning

Figure 29: Basic Safe Period Computation

The safe period formula in Eq. (34) assumes that the subscriber heads towards the alarm region in a straight line along the direction of the minimum Euclidean distance, an assumption that rarely holds true. One way to relax this stringent condition is to use the steady motion assumption: If the subscriber is heading towards the alarm region R, then the deviation in her motion direction is not likely to be extreme. Figure 29(b) shows a scenario where the bounded deviation in subscriber motion is taken into account for calculating average safe period for subscriber S approaching alarm region R. In order for the subscriber S to enter the alarm region R at some future time instant, the average angle of motion for the subscriber S over the safe period must lie between $-\theta_L$ and $+\theta_R$ (as shown in the

figure), which we refer to as alarm trigger angular range. Assume that the mobile subscriber heads towards the alarm region R in a direction at an angle θ to the minimum Euclidean distance vector; we refer to the distance from the subscriber position to the alarm region as the steady motion distance, denoted as $sm_{dist}(\theta)$. The steady motion-based safe period can be determined by $sm_{dist}(\theta)/f(V_S)$. Using the average steady motion distance obtained by computing $sm_{dist}(\theta)$ over all θ values ranging from $-\theta_L$ to $+\theta_R$, the steady motion-based safe period over the alarm trigger angular range can be calculated as,

$$sp = \frac{\int_{-\theta_L}^{+\theta_R} sm_{dist}(\theta)d\theta}{f(V_S)\int_{-\theta_L}^{+\theta_R} d\theta} = \frac{l+h}{f(V_S)(\theta_R+\theta_L)} \tag{35}$$

, where l, h denote the length and height of the spatial alarm region. The steady motion assumption provides a more realistic and optimistic measure for safe period computations compared to the minimum Euclidean distance approach.

4.2.2 Velocity Measurements

The use of maximum travel speed of the mobile client for the velocity function $f(V_S)$ carries both advantages and disadvantages. On one hand, the maximum travel speed can be set by pre-configuration based on a number of factors, such as the nature of the mobile client (car on the move or a pedestrian walking on the street), or the type of road used. On the other hand, maximum speed based estimation is often pessimistic, especially in the following two scenarios: (i) when the mobile client stops for an extended period of time, or (ii) when the mobile client suddenly turns onto a road with very low speed limit. Another issue related to the use of maximum speed of a mobile client for the velocity function f(VS) is related to alarm misses. The maximum velocity-based approach may fail to trigger alarms in cases where the maximum speed for the mobile subscriber increases suddenly. For example, a vehicle moving from a street onto a state highway would experience a sudden increase in its velocity, which may invalidate safe period computations. One way to address such sudden increase in velocity is to use dead reckoning techniques which require the mobile user to report to the server when her velocity increases over a certain threshold, as shown in Figure 29(c). The use of dead reckoning or similar techniques will allow the server to recompute the safe period for mobile client upon any significant velocity change. In Figure 29(c), the

mobile client keeps track of its predicted positions based on its maximum speed and its actual positions. As soon as the diffrence between the predicted position and the actual position exceeds a given threshold value (say δ), the client provides its current speed to the server.

4.2.3 Safe Period-based Alarm Evaluation

The safe period-based approach processes a spatial alarm in three stages. First, upon the installation of a spatial alarm, the safe period of the alarm with respect to each authorized subscriber is calculated. Second, for each alarm-subscriber pair, the alarm is processed upon the expiration of the associated safe period and a new safe period is computed. In the third stage, a decision is made regarding whether the alarm should be red or wait for the new safe period to expire. When compared to periodic alarm evaluation, the safe period approach for spatial alarm processing reduces the amount of unnecessary alarm evaluation steps, especially when the subscriber is far away from all her alarms. On the other hand, the main cost of the basic safe period approach described in this section is due to the excessive amount of unnecessary safe period computations, as the basic safe period approach performs safe period computation for each alarm-subscriber pair. One obvious idea to reduce the amount of unnecessary safe period computations is to group spatial alarms based on geographical proximity and calculate safe period for each subscriber and alarm group pair instead of each alarm-subscriber pair.

4.3 Alarm Grouping Techniques

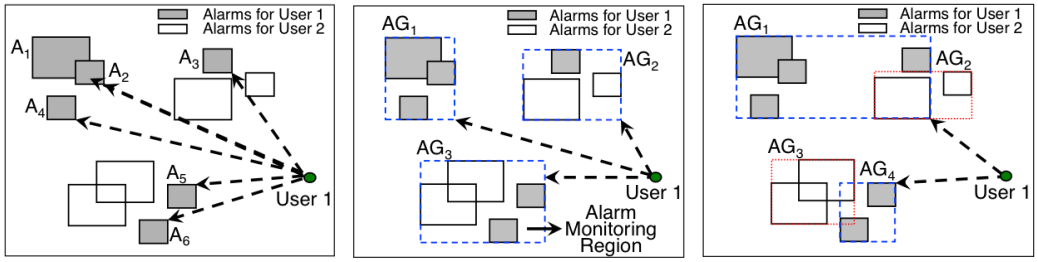
The basic premise behind alarm grouping is to reduce the number of safe period computations while ensuring no alarm misses. In this section, we present three alternative grouping techniques, each of which offers different degree of improvement for safe period computations. First, we group all alarms based on their spatial locality. Alternatively, we apply spatial locality based-grouping to alarms of each individual subscriber. Our experimental study shows that this approach is more effective. The third locality-based alternative is to employ the nearest alarms-based grouping, which is effective but costly when there are frequent alarm additions and removals.

4.3.1 Spatial Locality-based Grouping

Spatial locality-based (SL) grouping considers the set of alarms from all users and groups together the nearby alarms. This approach outperforms basic safe period alarm evaluation if each group has a large number of alarms belonging to the same subscriber. Figure 30(a) displays the alarm regions for a set of installed alarms. The alarms for user 1 are marked by shaded regions. Basic safe period evaluation computes the distance from each of the six alarms $\{A_i | 1 \leq i \leq 6\}$. In comparison, Figure 30(b) shows three groups derived from spatial locality-based grouping technique. We use a simple R-tree implementation in order to group alarms and identify the minimum bounding rectangles (MBRs) for alarm groups which are also referred to as alarm monitoring regions. Instead of computing distance for each alarm-subscriber pair, spatial locality-based grouping calculates the distance for each subscriber and alarm group pair. However, on entering a monitoring region the distance to all relevant alarms within the alarm group also needs to be computed. Despite this additional evaluation step, the number of safe period computations may be considerably reduced by grouping alarms according to spatial locality. Instead of six computations required by the basic safe period technique, only three computations need to be performed as all three alarm groups, $\{AG_i | 1 \leq i \leq 3\}$, contain alarms relevant to user 1. Further computations are dependent on the number of relevant alarms within the users' current alarm monitoring region. Even though this approach reduces the number of computations it requires considerable additional processing to determine the set of relevant alarm groups for each subscriber and the set of relevant alarms for each subscriber within an alarm group. The lack of subscriber-specificity in the underlying data structure, R-Tree, leads to retrieval of large number of unnecessary alarms. This technique proves to be efficient in presence of large number of public alarms as the effect of subscriber-specificity is reduced in this situation.

4.3.2 Subscriber-Specific Spatial Locality-based Grouping

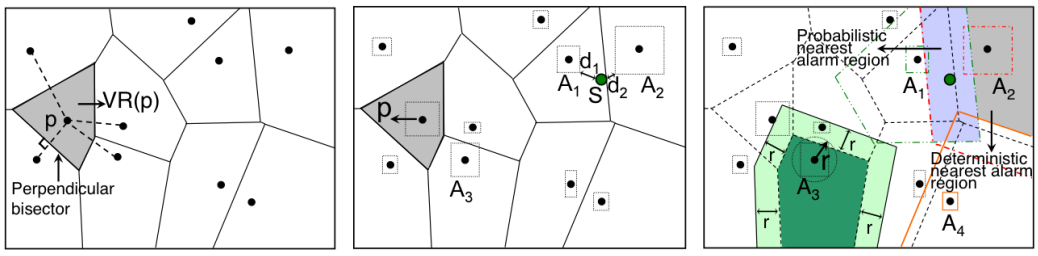
In contrast to spatial locality-based grouping, subscriber-specific spatial locality based (SSSL) grouping performs a two level grouping: the first level grouping is on all subscribers



(a) Per Alarm Safe Period (b) SL Grouping (c) SSSL Grouping

Figure 30: Alarm Locality-based Grouping

and the second level grouping is on spatial alarms relevant to each subscriber. We use a B-tree based implementation to speed up search on subscribers and an R-Tree implementation to capture spatial locality of alarms for each subscriber. The underlying data structure is a hybrid structure which uses a B-tree for subscriber search at the first level and an R-tree for subscriber specific spatial alarm search at the second level. Figure 30(c) shows an example of this grouping. Alarms installed by user 1 are grouped together in AG_1 and AG_4 and may be red only when the user is entering the MBRs of AG_1 or AG_4. Subscriber specific spatial locality-based grouping has two advantages over the previous approaches. First, the number of safe period computations is significantly reduced. Second, each alarm group contains alarms relevant to a single user, thus no irrelevant processing is performed. Our experimental results show that this approach is efficient in the presence of large number of subscribers and for large number of private and shared alarms.



(a) Voronoi Diagram (b) Voronoi Extension (c) Voronoi Extension

Figure 31: Nearest Alarms-based Grouping

4.3.3 Nearest Alarm-based Grouping

Nearest alarms-based grouping allows the system to perform one or only a few alarm checks dependent on the current subscriber position. The idea is to have each location on the map associated with the nearest spatial alarm region(s). In order to perform nearest alarms-based grouping we use an extension of the well known Voronoi diagram geometric structure [16]. The Voronoi diagram for a given set of points P in d-dimensional space R^d partitions the space into regions where each region includes all points with a common closest point P. The Voronoi region $VR(p)$ corresponding to any point $p \in P$ contains all points $p_i \in R^d$ such that,

$$\forall p' \in P, p' \neq p, dist(p_i, p) \leq dist(p_i, p') \tag{36}$$

Figure 31(a) shows the Voronoi diagram for a set of points in two-dimensional space R^2 with euclidean distance metric. The shaded area marks out the Voronoi region $VR(p)$ for the point p. In order to create a Voronoi diagram for spatial alarms we first represent each spatial alarm region R by its center point (x_{cr}, y_{cr}) and l, h representing the length and height of the alarm region. We consider the center point of each alarm region as a Voronoi site and create the Voronoi diagram as shown in Figure 31(b). But alarm regions may overlap with adjacent Voronoi regions as for alarm A_3 in the figure. Also, consider the subscriber S in the figure residing in the Voronoi region of alarm A_1. S is at a minimum Euclidean distance d_1 from the alarm region of A_1 and at a minimum Euclidean distance d_2 to the alarm region of A_2. Even though $d_2 < d_1$, A_1 is incorrectly identified as the nearest alarm on the basis of the underlying Voronoi diagram. In order to rectify this problem, we introduce an extension to the original Voronoi diagram by extending the boundary of each Voronoi region by the extension radius r associated with each point p where $r = \sqrt{(\frac{1}{2})^2 + (\frac{h}{2})^2}$. l, h denote the length and height of the alarm region associated with center point p. The extended Voronoi regions for alarms A_1, A_2, A_3 and A_4 are shown in Figure 31(c). Extending the Voronoi region boundaries leads to overlaps among neighboring Voronoi regions; subscribers inside overlapping regions (probabilistic nearest alarm region) may have more than one possible nearest alarm whereas subscribers inside non-overlapping

regions (deterministic nearest alarm region) can have only one nearest alarm.

Nearest alarm grouping is efficient for systems that have infrequent addition or removal of alarms and have no hotspots. However, it fails when there is frequent addition/removal of spatial alarms, since Voronoi diagrams need to be reconstructed each time an alarm is added or removed. In addition, high density of alarms in some areas may also lead to large overlaps among Voronoi regions, reducing the efficiency of this technique.

4.4 Experimental Evaluation

In this section, we report our experimental evaluation results. We show that our safe period-based framework and optimization techniques for spatial alarm processing are scalable while maintaining high accuracy.

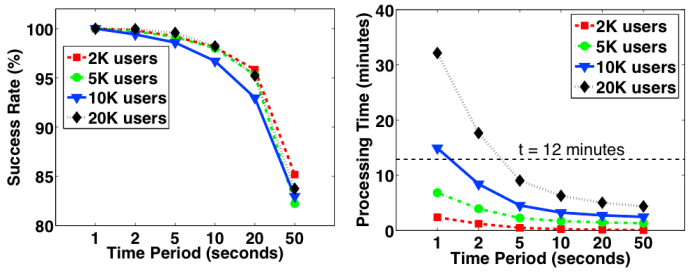
4.4.1 Experimental Setup

Our simulator generates a trace of vehicles moving on a real-world road network using maps available from the National Mapping Division of the U.S. Geological Survey (USGS [13]) in Spatial Data Transfer Format (SDTS [9]). Vehicles are randomly placed on the road network according to traffic densities determined from the traffic volume data in [47]. We use a map from Atlanta and surrounding regions of Georgia, which covers an area larger than 1,000 km^2, to generate the trace. Our experiments use traces generated by simulating vehicle movement for a period of fifteen minutes, results are averaged over a number of such traces. Default traffic volume values allow us to simulate the movement of a set of 20,000 vehicles. The default spatial alarm information consists of a set of 10,000 alarms installed uniformly over the entire map region; around 65% of the alarms are private, 33% shared and the rest are public alarms.

4.4.2 Experimental Results

The first set of experiments measures the performance of periodic alarm evaluation by varying the time period of updates and shows that this approach does not scale. The second set of experiments compares the basic safe period approach against periodic evaluation and

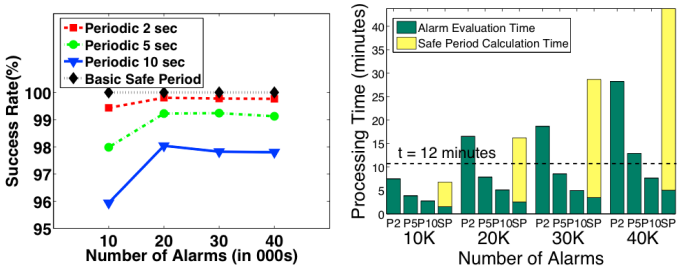
shows that safe period-based alarm processing offers higher success rate with lower evaluation time. The last set of experiments compares the performance of the various grouping-based optimizations against the basic safe period approach exhibiting the scalability of our grouping optimizations.



(a) Success Rate (b) Alarm Processing Time

Figure 32: Scalability with Varying Number of Users

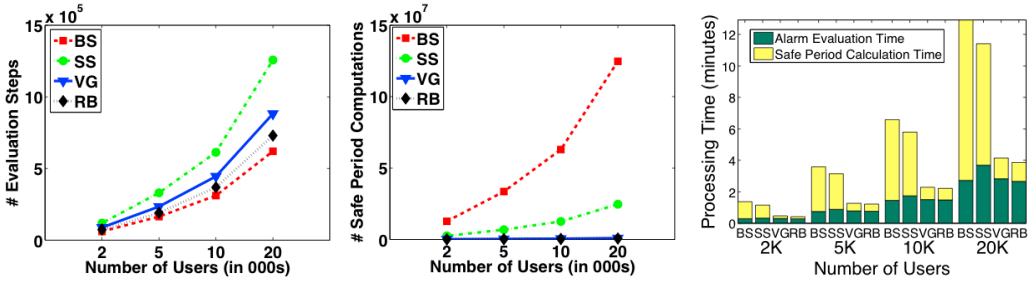
Scalability Problems of Periodic Alarm Evaluation Technique: In this first set of experiments, we measure the scalability of the periodic alarm evaluation technique with varying number of users. Figure 32 displays the results as we vary the number of users from 2K to 20K. The time period t_p for periodic alarm evaluation is varied from 1 second to 50 seconds. As can be seen from Figure 32(a), the success rate for alarm evaluation is 100% only if $t_p = 1$ second; for higher t_p success rate starts falling, even with $t_p = 2$ seconds the success rate does fall to 99.9% which may not be acceptable from QoS viewpoint as this translates to a significant number of alarm misses. The sequence of alarms to be triggered for 100% success rate are determined from a trace generated with highly frequent location updates for each user. The alarm processing time is plotted in Figure 32(b). Our traces are of fifteen minutes duration; considering that the system has around 80% of this time for processing spatial alarms we set the maximum processing time available to the system at $t = 12$ minutes as indicated by the horizontal dotted line in Figure 32(b). For 10K users the system is unable to process alarms at $t_p = 1$ seconds, thus failing to attain 100% success rate. For 20K users, this scalability problem becomes worse and the system is able to evaluate alarms only at $t_p = 5$ seconds. Thus, we conclude that periodic evaluation approach does not scale.



(a) Success Rate (b) Alarm Processing Time

Figure 33: Safe Period Optimization with Varying Number of Alarms

Performance Comparison with Basic Safe Period Approach: In this section, we compare the performance of basic safe period approach against periodic evaluation. We display the results for periodic approach with $t_p = 2$ seconds, $t_p = 5$ seconds, $t_p = 10$ seconds and the basic safe period optimization as discussed in Section 2 (P2, P5, P10 and SP in Figure 33(b)). Figure 33 displays the success rate and processing time as we increase the number of alarms from 10K to 40K. Figure 33(a) displays that the success rate is 100% for basic safe period approach and all periodic approaches miss at least a few alarm triggers. Figure 33(b) displays the alarm processing time for P2, P5, P10 and SP with varying number of alarms. The alarm processing time, as shown in Figure reffig:fig5b, displays the inability of our basic safe period approach to scale to large number of alarms. In presence of even 20K installed alarms, the approach has excessive safe period computation time which pushes the overall processing time beyond the 12 minute limit determined earlier. Our alarm grouping and subscriber mobility-based techniques provide optimizations to overcome this problem.



(a) Alarm Evaluation Steps (b) Safe Period Computation (c) Alarm Processing Time

Figure 34: Safe Period Optimization with Varying Number of Users

Scalability of Safe Period Evaluation Techniques: We now discuss the performance of the safe period optimization techniques to test the scalability of our framework. Figure 6 shows the number of alarm evaluation steps, number of safe period computations and the alarm processing time required by each approach: Basic Safe Period Optimization (BS), Subscriber-Specific Spatial Locality (SS), Voronoi Grid-Based (VG) and a Range-based Subscriber-Specific Grouping Optimization (RB). VG and RB approaches consider alarms only in the vicinity of the current subscriber position for safe period computation. Results for Spatial Locality-based grouping show expected trends but this approach has high overall processing time as the system needs to perform significant amount of computation to determine relevance of alarms/alarm groups for each subscriber. Hence, we exclude this approach from the results.

Figure 34(a) displays the number of alarm evaluation steps required by each approach. Basic safe period measures the safe period to each relevant alarm and uses this safe period to avoid further evaluations. As a result, this approach has to perform a low number of alarm evaluations but each evaluation step involves a very large number of safe period computations. Hence the number of safe period computations for this approach is extremely large (Figure 34(b)) which makes this approach overall computationally expensive as can be seen from the total alarm processing times in Figure 34(c). Subscriber-specific spatial locality grouping incurs a large number of alarm evaluation steps as can be seen from Figure 34(a). This approach first evaluates safe period for each alarm group; once the user enters an alarm monitoring region another evaluation step is required to determine the safe period to all alarms lying within the alarm monitoring region. Further, this approach needs to keep a check on subscriber position inside the alarm monitoring region and switch to per alarm group-based safe period computations once the subscriber moves outside the current alarm monitoring region. These additional evaluation steps imply that this approach will incur a larger number of alarm evaluation steps with each evaluation step requiring a small number of safe period computations: either for each alarm group or for all alarms lying within the current alarm monitoring region. Thus the number of safe period computations required by this approach is much lower than the basic approach despite the larger number

of alarm evaluation steps. Consequently, the overall processing time for SS is lower than the BS approach as can be seen from Figure 34(c). The VG and RB approaches lower the number of alarm evaluation steps by considering only alarms or alarm groups in the vicinity of the client. In this set of experiments, the RB approach considers alarms within a radius of 1,000m from the client position. VG approach overlays a grid with cell size 1,000m $\times$ 1,000m on top of the Voronoi extension and considers alarms only within the current subscriber grid cell. The number of evaluation steps for these approaches is still larger than the number of evaluation steps used by the basic approach as the safe periods computed by this approach may be lower than the safe period computed by the basic approach, in case no relevant alarms/alarm groups lie within the radius range or the current grid cell of the subscriber. However, each alarm evaluation step involves a very small number of safe period calculations leading to an extremely small number of safe period computations (in Figure 34(b) results for VG and RB are overlapping and values are much smaller than other two approaches). Consequently, the overall processing times for these two approaches are significantly lower than other approaches. From these results we can conclude that our safe period optimizations significantly aid the scalability of the system.

4.5 Related Work

An event-based location reminder system has been advocated by many human computer interaction projects [66, 87, 31, 69, 59]. Understandably, the primary focus of the work is from the point of view of the usability of such systems. None of these approaches deal with the system oriented issues which need to be resolved to make such systems feasible. In the realm of information monitoring, event-based systems have been developed to deliver relevant information to users on demand [11, 7]. In addition to monitoring continuously evolving user information needs, spatial alarm processing systems also have to deal with the complexity of monitoring user location data in order to trigger relevant alerts in a non-intrusive manner. Applications like Geominder [4] and Naggie [7] already exist which provide useful location reminder services using cell tower ID and GPS technology, respectively. Client-based solutions for spatial alarm processing should focus on efficiently evaluating spatial

alarms while preserving client energy. Our server-centric architecture makes it possible for users to share alarms and make use of external location information monitoring services which provide relevant location-based alerts. A server-centric approach is also essential for extending the technology to clients using cheap location detection devices which may not possess significant computational power.

4.6 Conclusion

We make two important contributions towards supporting spatial alarm based mobile applications. First, we introduce the concept of safe period to minimize the number of unnecessary alarm evaluations, increasing the throughput and scalability of the system. Second, we develop a suite of spatial alarm grouping techniques based on spatial locality of the alarms and motion behavior of the mobile users, which reduces the safe period computation cost for spatial alarm evaluation at the server side. We evaluate the scalability and accuracy of our approach using a road network simulator and show that the proposed safe period-based approach to spatial alarm processing offers significant performance enhancements for alarm processing on server side while maintaining high accuracy of spatial alarms.

CHAPTER V

SCALING SPATIAL ALARMS WITH MONDRIAN TREE INDEX

Spatial alarms are fundamental capability for location based advertisements and location based reminders. One of the most challenging problems in scaling spatial alarm processing is to compute alarm free regions (AFR) such that mobile objects traveling within an AFR can safely hibernate the alarm evaluation process until approaching the nearest alarm of interest. In this paper we argue that maintaining an index of both spatial alarms and empty regions (AFR) in the context of spatial alarm processing) is critical for scalable processing of spatial alarms. Unfortunately, conventional spatial indexing methods, such as R-tree family, k-d tree, Quadtree, and Grid, are not well suited to index empty regions. We present Mondrian Tree − a region partitioning tree for indexing both spatial alarms and alarm free regions. We first introduce the Mondrian tree indexing algorithms, including index construction, search, and maintenance. Then we describe a suite of Mondrian tree optimizations to further enhance the performance of spatial alarm processing. Our experimental evaluation shows that the Mondrian tree index outperforms traditional index methods, such as R-tree, Quadtree, and k-d tree, for spatial alarm processing.

5.1 Introduction

A spatial alarm is defined by three elements: *a future reference location* known as the alarm monitoring region and represented typically by a spatial region of interest, *an owner* who is the publisher of the alarm, and *the list of subscribers* of the alarm. In contrast to time-based alarms that remind us of the arrival of a *future reference time point*, spatial alarms remind us of the arrival of a *future reference location.*

We consider three categories of alarms: *private*, *shared* and *public*. *Private* alarms are installed and used exclusively by the publisher. *Shared* alarms are installed by the publisher with a list of authorized subscribers and the publisher is typically one of the subscribers. *Public* alarms are usually installed with the purpose of sharing them with all mobile users.

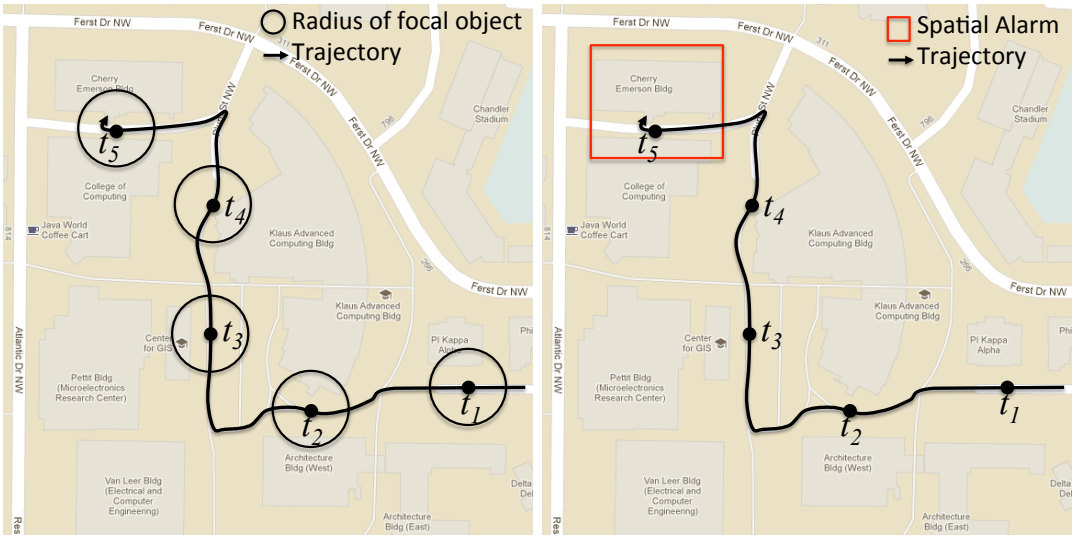
Public alarms can be useful means of informing subscribers about hazardous road situations, sever weather forecast, or delivering targeted advertisement.

Example 1: Location-based advertisements. North Face is an example of such services and it identifies the handsets of opted-in consumers within a certain radius of retail stores and sends text messages with discount offers [45]. This is an example of public alarms with subscriptions.

Example 2: Location-based reminders[87]. Alice sets a spatial alarm on a vitamin store in Lenox Square to *"remind her to pick up some vitamin products when she is within three miles of the store"*. This is an example of a private spatial alarm.

Spatial alarms are fundamentally different from spatial continuous queries. Figure 35 shows how spatial alarms are different from spatial continuous queries. Spatial continuous queries are targeted at mobile objects and events occur in the vicinity of the current location of the mobile user who issued the queries (i.e., the query focal object, represented as circles in Figure 35(a)). An example of spatial continuous query is "find a nearest restroom within 500 meters from the current location." As the user moves on the road, the spatial queries are always centered at the vicinity of the query focal object as shown in Figure 35(a). For example, we search spatial objects within 500 meters from the current location at t_1, t_2, t_3, t_4, and t_5. In contrast, spatial alarms are targeted at a future reference location of interest, instead of the current location of the mobile subscriber. Consider a spatial alarm such as "remind me to submit a petition for graduation near Cherry Emerson Building", which is represented as a red rectangle in Figure 35(b). When a user is moving on the road, the continuous query approach kepdf looking at the vicinity of the current location to see if the focal object is overlapping with the boundary of Cherry Emerson Building. However, the spatial alarm approach monitors the vicinity of the building if Alice is located near the building.

Clearly, if the user is far from the alarm monitoring region, all alarm evaluations are unnecessary and will not result in alarm notification. An ideal approach is to hibernate the spatial alarm application when the user is traveling locally and only trigger it when the user is approaching the Cherry Emerson Building within 200 meters. Thus, an important



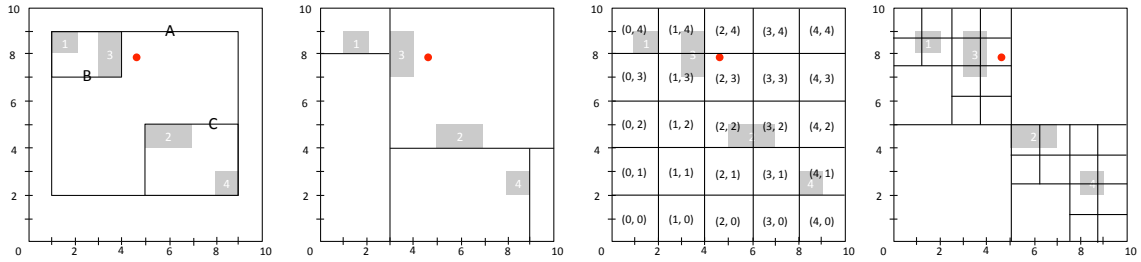
(a) Spatial Continuous Queries (b) Spatial Alarms

Figure 35: Comparisons between spatial Continuous Queries and spatial alarms

challenge in scaling spatial alarm processing is to compute alarm free regions such that mobile objects traveling within a rectangular region containing no spatial alarms can safely hibernate the alarm evaluation until approaching the nearest alarms of interest. For example, when user is at t_1 the mobile device starts to hibernating and wakes up at t_4 or t_5. We argue that in the context of spatial alarm processing, spatial alarms and alarm free regions are equally important and both should be treated as the first class citizen.

It is well known that maintaining an index of spatial data of interest is critical for retrieving those spatial data quickly. However, conventional spatial indexing methods, such as R-tree [48], k-d tree [21], Grid [73], and Quadtree [37], are not well suited to index empty regions. Figure 36 shows how conventional indexing methods index spatial data of interests, represented as gray rectangles with numbers from 1 to 4 (spatial alarms in this context). The user's current location is depicted as a red dot. Note that all structures except Quadtree and Grid do not index empty regions (the non-gray regions.) Given the current user location, R-tree and k-d tree know that the user is not inside of any spatial alarms but do not know what the size of empty region where user can move around without worry of entering spatial alarms. Grid knows that user is in the cell (2,3), which is empty, and the size of the cell. However, if a user moves to a cell with a spatial alarm such as

(1, 3), then Grid cannot tell what the size of empty region without computing the empty region. Like Grid, Quadtree sometimes knows the size of empty region. Given the current location, however, Quadtree cannot tell the size of empty region without AFR computation.



(a) R-tree (b) k-d tree (c) Grid (d) Quadtree

Figure 36: Indexing spatial alarms using conventional spatial indexing methods

It is not impossible to index empty regions for these conventional index structures. For R-tree and k-d tree we can manually create small empty regions and add them into the index. Then it becomes the Mondrian tree. For Grid the cell size should be smaller enough so that one of the cell boundaries coincides the borders of spatial alarms. This smaller cell size causes two problems: (a) bigger storage and (b) frequent crossing cells. We can further partition the cell of Quadtree so that the cell boundaries touch alarm boundaries, we immediately know where the empty regions are. Then like Grid, we face same issues. This example illustrates that if we want to index both spatial alarms and the empty regions, the conventional indexes are no longer suitable since those indexes by design work effectively only when some sub-regions in the universe of discourse containing the spatial objects of interest need to be indexed.

In this paper we present the design of Mondrian tree, a region partition index for both spatial alarms and alarm free regions (AFR). We first describe the Mondrian tree index construction, search, and maintenance algorithms. Then we describe a suite of optimizations to further enhance the performance of Mondrian indexing and spatial alarm processing. Mondrian tree index has two unique features. First, it utilizes the pre-computation and indexing of empty regions to avoid on-the-fly computation of alarm free regions based on the motion behavior of mobile subscribers. Second, it incorporates a suite of locality-aware

and motion-aware optimizations to further minimize the amount of wakeups and the number of region-crossing checks to be performed at mobile clients. We conduct a set of extensive experimental evaluations and show that the Mondrian tree indexing offers fast spatial alarm processing, and it significantly outperforms existing spatial indexing methods, such as R-tree, k-d tree, and Quadtree, which compute alarm free regions dynamically based on the motion behavior of mobile users.

5.2 Overview and Related Work

A critical challenge for efficient processing of spatial alarms is to determine *when to evaluate* each spatial alarm, while ensuring the two demanding objectives: *high accuracy*, which ensures zero or very low miss rate of spatial alarms, and *high efficiency*, which requires highly efficient processing of spatial alarms.

Periodic evaluation can be performed for spatial alarms by checking whether a mobile subscriber is entering the spatial alarm on every pre-defined time interval. High frequency is essential to ensure that none of the alarms are missed. Though periodic evaluation is simple, it can be extremely inefficient due to frequent alarm evaluation and the high rate of irrelevant evaluations [72].

Similarly, **processing spatial alarms upon location updates** of mobile users is equally incompetent and wasteful due to the specific characteristics of spatial alarms. For example, assume that the user is currently at t_1, 10 miles away from her spatial alarm as shown in Figure 35(b). Then it is unnecessary to evaluate those spatial alarms upon her location updates when she approaches at t_4 or t_5 [18].

Safe regions are popular techniques for continuous spatial query processing [39, 85, 80, 50]. The safe region of an object o is dynamically computed at the server based on the set of queries such that the current results of all queries remain valid as long as o is residing inside its safe region. Computing safe region takes $O(n^2)$ for n queries [80]. Although [18] extends the safe regions to spatial alarm processing, the high cost of dynamic safe region computation remains to be a challenging problem. As the mobile client moves, the server needs to re-compute its new safe region continuously, which can be expensive. Figure 37

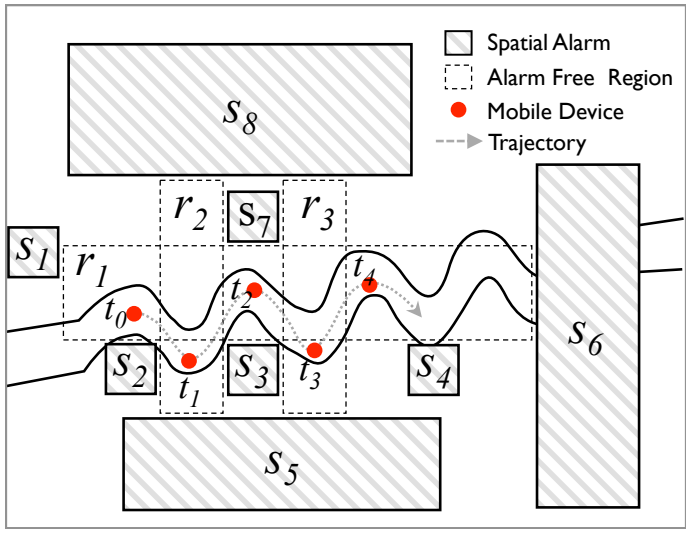


Figure 37: Safe regions computed at time t_i

shows spatial alarms installed near the Lombard street, San Francisco. At t_0, the safe region is r_1, the wide rectangle, because it is the largest rectangle that does not overlap with any spatial alarms. At t_1, the client exists r_1 and the server computes a new safe region, r_2, the tall rectangle. At t_2, however, the client gets out of r_2 and the server computes a new safe region which is the same as r_1. The computation of the same safe region, r_1, occurs at t_0, t_2 and t_4 as the client moves along the Lombard street. This example scenario shows that although the safe region approach reduces the amount of unnecessary alarm processing, it also introduces a fair amount of unnecessary safe region computation.

Bearing these issues in mind, we present the Mondrian[1] tree indexing structure, which partitions the universe of discourse into smaller regions of two types: spatial alarm regions and empty regions. We call empty regions Alarm Free Regions (AFR) because there is no alarm inside of the region. With Mondrian indexing, it takes $O(\log n)$ for searching the AFR of the mobile client, which is much more efficient compared to $O(n^2)$ for computing safe region on the fly. To our best knowledge, Mondrian tree is the first index structure to partition the universe of discourse into small regions containing objects of interest (spatial alarms in our context) and empty regions and index them all.

[1]The name Mondrian is named after Piet Mondrian because region partitioning resembles his "Composition with red, yellow blue and black".

5.3 Basic Mondrian Tree Index

In this section we introduce the basic Mondrian tree index, including data structure, batch index construction, search, and insertion algorithm. Although Mondrian tree index is a general indexing structure and can be used to index any type of spatial objects, in this paper we will introduce it in the context of spatial alarms.

5.3.1 Definitions and Notations

Given a universe of discourse containing a set of m spatial alarms $\{o_j | 1 \leq j \leq m\}$, the corresponding Mondrian tree has a set of n nodes: $\{v_i | 1 \leq i \leq n\}$. Each v_i stores the partitioned disjoint rectangular region, which is either an empty region or a region containing o_j. Without loss of generality, we denote each spatial alarm o_j by its minimal bounding rectangle MBR_j. The leaf nodes of a Mondrian tree are either MBR(spatial alarms) or AFR(empty regions). The internal nodes of the Mondrian tree contain a set of pointers to its children nodes. We represent and store each spatial region in Mondrian tree as points using a point transformation scheme [84]. A k-dimensional rectangle shaped region with each side parallel to one dimensional axis is represented by a $2k$-dimensional point. For example, a 2-dimensional rectangular region r is defined by its lower left corner and upper right corner as a 4-dimensional point r:

$$r = (p_0, p_1, p_2, p_3),$$

where (p_0, p_1) is the lower left corner and (p_2, p_3) is the upper right corner of the rectangle. From now on, we use $r[i]$ to denote the $(i+1)$-th coordinate value of a 4-dimensional point r. For example, $r[2]$ is p_2. If r is an array, then $r[i]$ is the $(i+1)$-th item. Each node v_i consists of 5 components:

$$v_i = (R_i, \text{OID}_i, \text{SI}_i, K_i, \text{CHILD}_i).$$

R_i is a disjoint rectangular region that v_i represents and expressed as a $2k$-dimensional point as explained above. OID_i is a set of identifiers of the spatial alarms, which overlaps with the rectangular region R_i. Each non-leaf node is split into two children nodes. SI_i, a

split index, and K_i, a key, are used for performing node split of internal node v_i. SI_i decides which axis to use for partitioning the node v_i and K_i is the value of the split line. SI_i is an integer value between 0 and $2k-1$ and is defined as:

$$\text{SI}_i = d_i \ mod \ 2k,$$

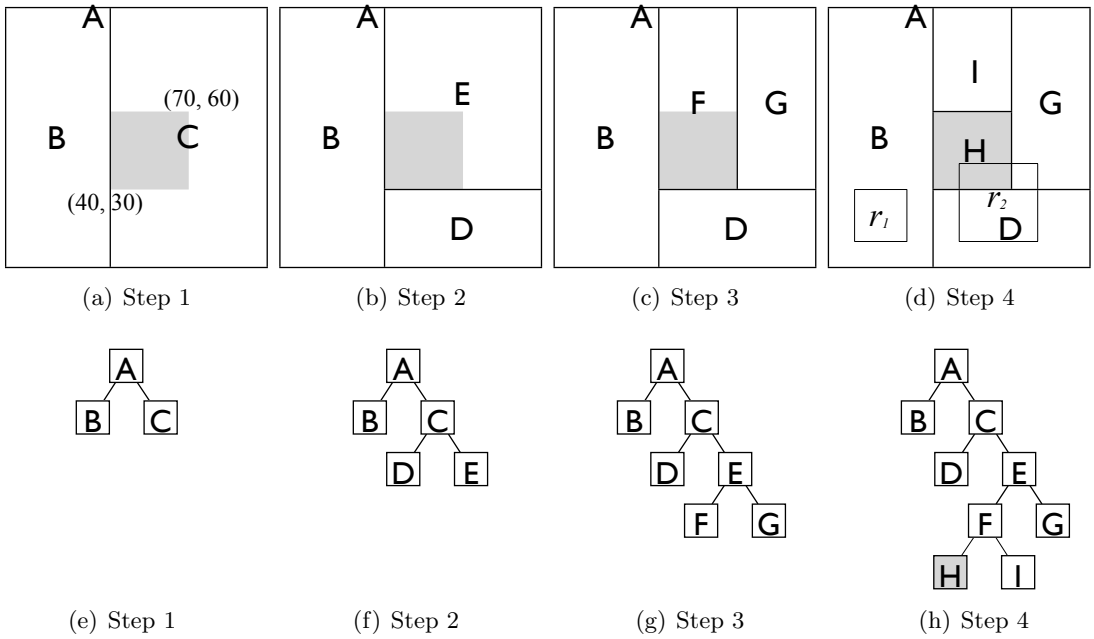
where d_i is depth of v_i. If SI_i is zero or an even integer value, then we split the internal node v_i along the line parallel to x axis, otherwise, along the line parallel to y axis. When splitting on v_i containing o_j, K_i is $\text{MBR}_j[\text{SI}_i]$. CHILD_i is a set of pointers to either null or the children nodes of v_i. Leaf nodes do not have K_i because they are either empty regions or MBR_j and thus nonsplit nodes. We set the split index SI_i of leaf nodes to be -1. For the root node, v_0, its depth d_0 is 0. Given k as 2, SI_0 is 0, which is computed by $(0 \ mod \ 2)$.

5.3.2 Region Partitioning through Node Split

Given a node v_i, partitioning a node is processed using SI_i, K_i, and o_j. For each o_j, we have $\text{MBR}_j = (p_0, p_1, \ldots, p_{2k-1})$. Assuming that the value of the split index SI_i is set to 0. Then $K_i = \text{MBR}_i[\text{SI}_i]$ and $\text{MBR}_i[\text{SI}_i] = \text{MBR}_i[0] = p_0$, and we split the node v_i along the line parallel to the y axis represented by $x = p_0$. Now the region of v_i is split into two smaller disjoint regions, denoted by $\text{CHILD}_i[0]$ and $\text{CHILD}_i[1]$. One of the two children regions contains o_j. Assuming that it is $\text{CHILD}_i[1]$, then the node split process for v_i repeats in the region of $\text{CHILD}_i[1]$. In each iteration SI_i and K_i are computed again, and the node split of v_i is performed along the line parallel to the x or y axis determined by SI_i and K_i. This node split process iterates until every p_l $(0 \leq l \leq 2k-1)$ in R_i is examined.

Figure 38 shows the example of region partition and Mondrian tree. Table 5 shows how data is stored in the Mondrian tree. Node v_0 is the root node that covers the rectangle region A represented by $(0, 0, 100, 100)$. Initially we set SI_0 as 0. MBR_0 for the spatial alarm o_0 is $(40, 30, 70, 60)$. In step 1, we compute the key K_0 by $\text{MBR}_0[\text{SI}_0]$, which is 40. Then v_0 is divided into two children along the line parallel to y-axis represented as $x = 40$. Now v_0 has two children v_1 for rectangle B and v_2 for rectangle C as shown in Figure 38(a). R_1 for v_1 is $(0, 0, 40, 100)$, which is computed from (0, 0, K_0, 100) and R_2 for v_2 is (40, 0, 100, 100), computed from $(K_0, 0, 100, 100)$. In step 2, v_2, rectangle C, is chosen to be partitioned

because o_0 intersects with v_2. v_2 is divided into v_3, rectangle D, and v_4, rectangle E as shown in Figure 38(b). In step 3, v_4 is chosen and partitioned into v_5, rectangle F, and v_6, rectangle G. At last, v_5, rectangle F, is divided into v_7, rectangle H, and v_8, rectangle I. As a result, this example Mondrian tree indexes both the spatial alarm represented by a gray rectangle H using index node v_7 but also four empty regions denoted by B (v_1), D (v_3), G (v_6), and I (v_8).



(a) Step 1 (b) Step 2 (c) Step 3 (d) Step 4

(e) Step 1 (f) Step 2 (g) Step 3 (h) Step 4

Figure 38: Partitioning Nodes

Table 5: Data stored in the Mondrian tree

v_i	MBR_i	SI_i	K_i
v_0 (A)	$(0, 0, 100, 100)$	0	40
v_1 (B)	$(0, 0, 40, 100)$	1	N/A
v_2 (C)	$(40, 0, 100, 100)$	1	30
v_3 (D)	$(40, 0, 100, 30)$	2	N/A
v_4 (E)	$(40, 30, 100, 100)$	2	70
v_5 (F)	$(40, 30, 70, 100)$	3	60
v_6 (G)	$(70, 30, 100, 100)$	3	N/A
v_7 (H)	$(40, 30, 70, 60)$	0	N/A
v_8 (I)	$(40, 60, 70, 100)$	0	N/A

Algorithm 8: PointSearch

Input: Node v_i and current location p.
Output: A leaf node v that contains p.

```
1 if v_i = leaf then
2     return v_i
3 end
4 Dim ← SI_i mod k    // Determine split line dimension; 0 for x or 1 for y
5 if p[Dim] < K_i then
6     return PointSearch(Child_i[0], p)    // p is in left or bottom of split
      line
7 else
8     return PointSearch(Child_i[1], p) // p is in right or top of split line
9 end
```

5.3.3 Search Operation

In spatial alarm processing, we use the Mondrian tree to find whether a mobile user enters a spatial alarm region or stays in an alarm free region which is an empty region. We refer to this search operation as a point search. In addition, we also need a region search operation that can find which index node overlaps with a given rectangle region.

Point search over the Mondrian tree is to find a leaf node node v_i that contains a point p. For example, in the context of two dimensional data space, the point search can be used to answer a query like "What is the smallest rectangular region that contains the point (x, y)?" or "Does the mobile user at position (x, y) enter a spatial alarm region?".

Given a point p in k dimensional space, PointSearch in Algorithm 8 is the point search algorithm that takes the Mondrian tree and the point p as input and outputs the leaf node (either a spatial alarm region or an empty region) in which p resides. The algorithm starts the search from the root node. For each node v_i, PointSearch determines which dimension is used for v_i to split the node (line 4). For this dimension, we compare coordinate of p with the key, K_i, associated with v_i. If $p[Key]$ is greater than or equal to K_i, then PointSearch searches the right subtree because p is located on right side (or upper side) of the split axis. Otherwise, the left subtree will be searched. PointSearch repeats this process iteratively until it reaches the leaf node that contains p.

The time complexity of the point search operation on a Mondrian tree is proportional to

the height of the tree given its binary search feature. The average height of a binary search tree with n nodes is $\log n$. Therefore, the time complexity of PointSearch is also $O(\log n)$. For the skewed Mondrian tree, the height might be n. Then the worst time complexity of PointSearch is $O(n)$.

Region search algorithm RegionSearch finds all leaf nodes in the Mondrian tree, which overlap with a given rectangle r. Concretely, RegionSearch first examines the root node v_i to find if one of its two children nodes overlaps with r. Let c be SI_i *mod* k and c be $(c + k)$ *mod* k. Unlike PointSearch, RegionSearch compares v_i's key with an interval. The interval is given by an upper bound and a lower bound in the c-th dimension of the given rectangle r, denoted by $\text{Min}(r[c], r[c])$ and $\text{Max}(r[c], r[c])$. If K_i is greater than or equal to the upper bound, then the left children $\text{Child}_i[0]$ will be taken as the input and the algorithm RegionSearch starts the new iteration with $\text{Child}_i[0]$ and r. If K_i is smaller than or equal to the lower bound, then the right children $\text{Child}_i[1]$ will be taken as the input and the algorithm RegionSearch starts the new iteration with $\text{Child}_i[1]$ and r. Otherwise, both children of v_i overlap with r. Thus, both RegionSearch($\text{Child}_i[0]$, r) and RegionSearch($\text{Child}_i[1]$, r) are invoked. This process repeats until the leaf nodes are reached.

Consider the Mondrian tree in Figure 38(h) and a rectangle $r_1(10, 10, 20, 30)$ as shown in Figure 38(d), the algorithm RegionSearch returns v_1 representing region B. Given a rectangle r_2(50, 10, 70, 40), the algorithm RegionSearch returns three leaf nodes, v_3, v_6, and v_7, which represent rectangle regions D, G, and H, all overlapping with r_2. Concretely, r_1's x interval is (10, 20) and y interval is (10, 30). Given that K_0 as 40, x's upper bound, 20, is smaller than K_0. Therefore the rectangle r_1 is overlapping with $\text{Child}_0[0]$, which is the rectangle region B and located at left of the split line. Because B is the leaf node, RegionSearch finishes returning B. Similarly, for r_2 x interval is (45,65) and y interval is (10,40). Thus, K_0 is smaller than the x's lower bound of r_2, which is 45. That is r_2 is overlapping with v_2. We compare y interval with K_2 and K_2 is staying between y interval. Therefore RegionSearch launches two RegionSearch, one with v_3 and the other with v_4. The first launch finishes because v_4 is the leaf node. The second launch also launches

Algorithm 9: RegionSearch

Input: Node v_i and rectangular region r.
Output: A set of leaf nodes overlapping with r.

```
1  if v_i = leaf then
2      return v_i
3  end
4  c ← Si_i mod k                          // Determine index of one end of range
5  c' ← (c + k) mod k                      // Determine index of the other end of range
6  min ← Min(r[c], r[c'])                  // Left end of range
7  max ← Max(r[c], r[c'])                  // Right end of range
8  if max ≤ K_i then
9      return RegionSearch(Child_i[0], r)  // r is in left of split line
10 end
11 else if K_i ≤ min then
12     return RegionSearch(Child_i[1], r)  // r is in right of split line
13 end
   // r is overlapping with split line
14 nodes_left ← RegionSearch(Child_i[0], r)
15 nodes_right ← RegionSearch(Child_i[1], r)
16 return nodes_left ∪ nodes_right
```

two RegionSearch. We keep going down until all RegionSearch reach leaf nodes. As a result we have D, G, and H.

Similar to a point query, RegionSearch finishes if it arrives at the leaf node. Therefore, RegionSearch also takes $O(\log n)$ on average and $O(n)$ for the worst case.

5.3.4 Insertion

The algorithm of inserting a new spatial alarm o_j to a Mondrian tree starts from the root node and find a node v_t using RegionSearch. v_t is either a non-leaf node whose children both overlap with Mbr_j or a leaf node. Then we consider three cases:

(i) $\text{Mbr}_j \cap R_t = \text{Mbr}_j$. Mbr_j is smaller than and fully contained in R_t of a leaf node v_t. Then we split v_t into $\text{Child}_t[0]$ and $\text{Child}_t[1]$, such that only one of them fully contains Mbr_j, say $\text{Child}_t[1]$. Now Insert is invoked with $\text{Child}_t[1]$ and o_j (Lines 5-10).

(ii) $\text{Mbr}_j = R_t$. Mbr_j is the same as R_t. Then we add the identifier of oj into Oid_t and stop the algorithm (Line 3).

(iii) Otherwise, v_t is a non-leaf node and MBR_j is overlapping with both children nodes of v_t. Then we split MBR_j into two disjoint partitions along the line computed by SI_t and K_t. Now the insertion algorithm is invoked to insert two partitions of o_j into the two children nodes of v_t one at a time (Lines 13-15).

We provide a sketch of the pseudo code in Algorithm INSERT. Given a Mondrian tree of n nodes, the time complexity of INSERT is $O(\log n)$. INSERT consists of two parts: (1) find v_t and (2) partition v_t or MBR_j. First part takes usually $O(\log n)$. The second part takes $O(1)$.

Algorithm 10: INSERT

```
   Input: Node v_i and spatial alarm o_j.
 1 if v_i is leaf then
 2     if MBR_j = R_i or OID_i ≠ null then
 3         Add O_j.ID into OID_i                                  // case ii
 4     else
 5         splitNode(v_i)                                         // case i
 6         if CHILD_i[0].R contains MBR_j then
 7             INSERT(CHILD_i[0], o_j)
 8         else
 9             INSERT(CHILD_i[1], o_j)
10         end
11     end
12 else
13     (o_j.left, o_j.right) = splitAlarm(o_j)                    // case iii
14     INSERT(CHILD_i[0], o_j.left)
15     INSERT(CHILD_i[1], o_j.right)
16 end
```

Although Mondrian tree is a memory-based tree, it can be extended to a tree with page-oriented storage like hard disks. Concretely, if the capacity of memory is n nodes, then those n nodes are stored in the memory and the remaining nodes are stored on external pages. We adopt this structure from LSD tree [51]. In this paper, we briefly describe basic ideas of using external pages. For more detailed algorithm, you can refer the LSD paper. Figure 39 shows the combination of memory and external storage. If the size of M exceeds n, then the subtree of M is written into an external page. If the height of a subtree in external pages exceeds h_p, which is set by the system, then the external page is split into two pages.

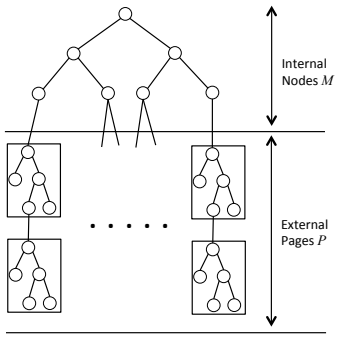


Figure 39: Memory and external storage

5.3.5 Deletion

The deletion algorithm consists of two steps. The first step is to find nodes that have the alarm to be deleted. Given the alarm, a set of nodes that overlaps with the alarm can be found if we maintain a hash table for a pair of alarm identification and a set of nodes that contains the alarm. Then for each found node v, we remove the alarm from v. The second step handles the merge operation. For example, in Figure 38(h), if we remove the alarm from H, then we can remove all nodes except A. As a result, this delete operation is the same as merge operation.

Algorithm 11: DELETE

```
   Input: A set of nodes overlapping with a spatial alarm o_j
1  foreach Node v in set of nodes do
2      v.removeID(o_j.ID)
3      MERGE(v)
4  end
```

Algorithm 12: MERGE

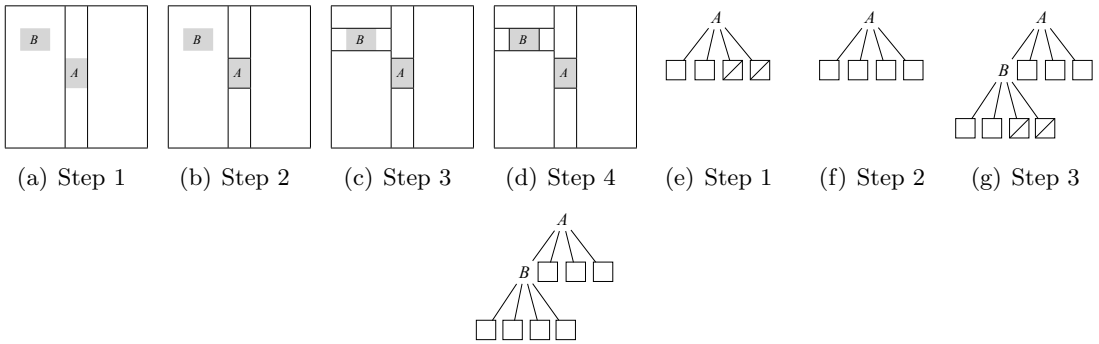
```
   Input: Node v
1  p ← v.getParent()
2  if p ≠ NULL and p's children has no alarms then
3      p.removeChildren()
4      MERGE(p)
5  end
```

5.4 Mondrian: An Optimized Tree*

The basic Mondrian index does not guarantee the balance of the tree. Once a node is partitioned into two disjoint smaller regions, one of them is not touched and the other is selected for further partition. Therefore the basic Mondrian index cannot guarantee the balance of the tree. Also the skewness of the tree is affected by the insertion order. In this section we introduce Mondrian*, an optimized Mondrian index structure that produces a more balance tree.

The Mondrian* tree also has similar **data structure** except that each node has four children instead of two, denoted by $\text{CHILD}_i[0]$, $\text{CHILD}_i[1]$, $\text{CHILD}_i[2]$, and $\text{CHILD}_i[3]$. Each rectangular region is stored at each node using four pointers instead two in the basic Mondrian tree. Given v_i, CHILD_i has pointers to left, right, bottom and top of MBR_i. Thus, the Mondrian* tree has four keys in each node because we split the space by four lines.

Figure 40 shows an example of how the Mondrian* partitions the space. For the chosen spatial alarm, say A, the Mondrian* partitions the space by drawing two vertical lines along the alarm as shown in Figure 40(a). Now we set $\text{CHILD}_0[0]$ and $\text{CHILD}_0[1]$ as shown in Figure 40(e). A box with a diagonal line means that node is null. Then two horizontal lines are drawn as shown in Figures 40(b) and we set $\text{CHILD}_0[2]$ and $\text{CHILD}_0[3]$ as shown in Figure 40(f). When B is inserted, then we put B on $\text{CHILD}_0[0]$ because B lies on left of A. Then we perform the same region partition operation on B as shown in Step 3 and Step 4 of Figure 40.



(a) Step 1 (b) Step 2 (c) Step 3 (d) Step 4 (e) Step 1 (f) Step 2 (g) Step 3

(h) Step 4

Figure 40: Region Partition in Mondrian*

Mondrian* Index Batch Construction. Given a set of n spatial alarms, the worst case in index construction occurs when every spatial alarm $o_j (1 \leq j \leq m)$ is inserted as a child at the lowest leaf node. One advantage of batch index construction is to utilize the prior knowledge on the distribution of spatial alarms to avoid the extreme skewedness of the tree. In this section we describe the batch construction method of Mondrian* index such that no subtree has more than one half of the nodes in the Mondrian* tree. For presentation brevity, we use k as 2 for two dimensional space as our context.

The first step is to sort the set of n spatial alarm objects in x coordinate and secondly in y coordinate. Given a sorted list of spatial alarm objects, we choose the spatial alarm object that is the medium of the ordered list as the root node of the Mondrian* tree, denoted by o_{root}. We insert this o_{root} first by performing the region partition as discussed above (recall example in Figure 4). The four children of o_{root} are created and denoted as $\text{CHILD}_{root}[0]$ (left), $\text{CHILD}_{root}[1]$ (right), $\text{CHILD}_{root}[2]$ (bottom), and $\text{CHILD}_{root}[3]$ (top). Furthermore, the remaining spatial alarm objects in the ordered list are regrouped into four sub-collections, each of which will be inserted into one of four children of the root. Those spatial alarms that are ranked before the chosen root object will be placed with $\text{CHILD}_{root}[0]$ (left) or $\text{CHILD}_{root}[2]$ (bottom) and the remaining spatial alarms that are ranked after the chosen root object will be indexed through $\text{CHILD}_{root}[1]$ or $\text{CHILD}_{root}[3]$. This process iterates recursively until all alarm objects in the ordered list are inserted in the Mondrian* tree. Clearly, this batch construction process ensures that no subtree can possibly contain more than half of the total number of nodes.

The time complexity of this algorithm is $O(n^2 \log n)$, given the ordering step on n spatial alarm objects takes $O(n \log n)$, the selection of the median requires $O(1)$, and INSERT takes $O(n)$.

Figures 41(a) and 41(b) show the result of region partitioning by basic Mondrian tree and Mondrian* tree after the batch index construction over six spatial alarms. Figures 41(c) and 41(d) shows the corresponding Mondrian and Mondrian* tree. Although the spatial alarm D is chosen as the root node for both basic Mondrian and Mondrian*, the region partitioning result for Mondrian* is quite different than basic Mondrian tree. So is the

index structure. This example illustrates that Mondrian* is much more balanced compared to the basic Mondrian index and thus offers much higher efficiency in terms of search and insertion.

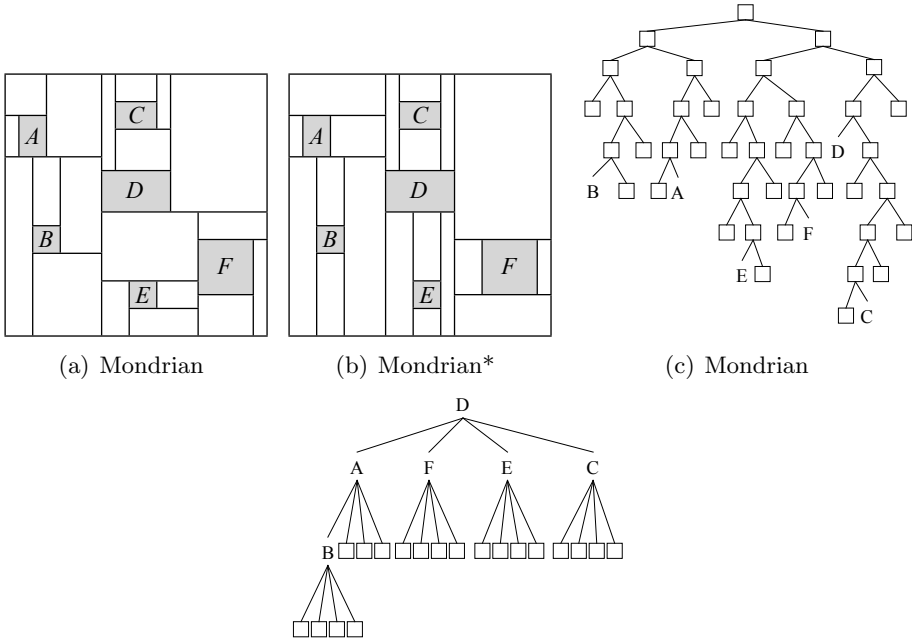


Figure 41: Region Partitioning: An Example

Search algorithm for Mondiran* works in a similar way as the basic Mondrian search. The only change that we need to add to the Mondrian* search is the index key comparison part since each node in Mondrian* has four keys, $K_i[0]$, $K_i[1]$, $K_i[2]$, and $K_i[3]$. Given a point p and a node v_i in Mondrian*, the point search algorithm uses DIRECTION(p, v_i) to compute and return a pointer to one of v_i's four children, which contains p, by comparing p's coordinate value with K_i. For example, if DIRECTION(p, v_i) returns 2 then we visit CHILD$_i$[2].

5.5 Spatial Alarm Evaluation

We have presented Mondrian tree and Mondrian* tree for indexing spatial alarms and empty regions. Intuitively, we can treat each empty region as an alarm free region (AFR) such

that when a mobile user travels inside an AFR, the alarm evaluation is hibernated, saving both energy consumption at the mobile client and the alarm evaluation cost at the server. However, as shown in Figure 41, an empty region in Mondrian tree may not be the best AFR for a mobile user, especially when multiple AFRs are adjacent to one another. In this section we describe the best strategies for evaluating spatial alarms using Mondrian tree indexes, including alarm free period, patched and trimmed AFR, motion-aware AFR and distributed Mondrian indexing scheme.

In principle, a spatial alarm should be evaluated in three steps. First, we need to determine what type of events should activate the spatial alarm evaluation process. Second, the server needs to find out the list of alarms to be evaluated upon the occurrence of the alarm events. The shorter this list is, the more efficient the spatial alarm evaluation will be. Third, the server executes the action component of those spatial alarms whose alarm conditions are evaluated to be true.

As discussed in Section 5.2, periodic evaluation is extremely inefficient due to frequent alarm evaluation and high rate of irrelevant evaluations. Although using the location update of a mobile user as the alarm evaluation event seems appealing, and it is independent of the concrete location update strategies, such as periodic, dead-reckoning or others [77], we have pointed out in Section 5.2 that many location update events are not suitable as the alarm evaluation events. First, not all location updates of a mobile user will lead to a successful evaluation of her spatial alarms, especially when she travels in the spatial area that does not contain any of her spatial alarms. Second, location updates of a mobile user will have zero probability of leading to successful evaluation of those spatial alarms that are not owned or subscribed by this mobile user. For instance, Bob's private spatial alarms are indifferent to the location updates of Alice.

To address the first issue, we promote the use of alarm free regions such that no spatial alarm evaluation will be activated when a mobile user travels inside an alarm free region. This can significantly reduce the frequency and overhead of spatial alarm evaluation. To address the second issue, the server needs to find out the list of alarms to be evaluated upon the occurrence of the alarm events. The shorter is this list, the more efficient is the

spatial alarm evaluation. This motivates us to design the distributed Mondrian tree index structure.

5.5.1 Alarm Free Period

An obvious idea for evaluating spatial alarms efficiently is to incorporate the spatial locality of the alarms and the motion behavior of mobile objects through alarm free regions. We have discussed in Section 5.2 that dynamic computation of alarm free regions is expensive due to unnecessary and possibly duplicate AFR computation. Given that Mondrian tree approach indexes both spatial alarm regions and empty regions, it is intuitive to use empty regions as alarm free regions (AFRs). The only cost for using AFRs is time to lookup the leaf node instead of computing AFRs. Once AFR is acquired, then the client needs to check if it is still inside of AFR. We introduce the concept of alarm free period (AFP) as a basic approach to assist a mobile user to determine when to check whether she moves outside of her current AFR. An important property of AFP is that it should avoid missing alarms or minimize the alarm miss rate.

5.5.1.1 Basic AFP

Given a mobile client m and an alarm free region AFR_m, the AFP_m is the shortest travel time for m to arrive at the closest border of its current alarm free region AFR_m. During AFP_m, m's alarm evaluation service can enter a sleep (hibernate) mode.

Two main factors that impact on the computation of the AFP_m are the velocity of m, say V_m, and the shortest distance from the current position of m to the closest border of AFR_m, say $MinDist(m, \text{AFR}_m)$. Thus, the AFP_m can be computed as follows:

$$\text{AFP}_m = \frac{MinDist(m, \text{AFR}_m)}{V_m} \tag{37}$$

One caveat with Equation 37 is that it assumes that the mobile subscriber m moves in a straight line from her current location to the closed border of AFR_m. It is, however, not a realistic assumption in real life. For example, it is highly likely that the mobile user m is moving towards a direction that is opposite of the closest border of its current AFR_m. The steady motion assumption is specifically true when mobile users move on the road networks.

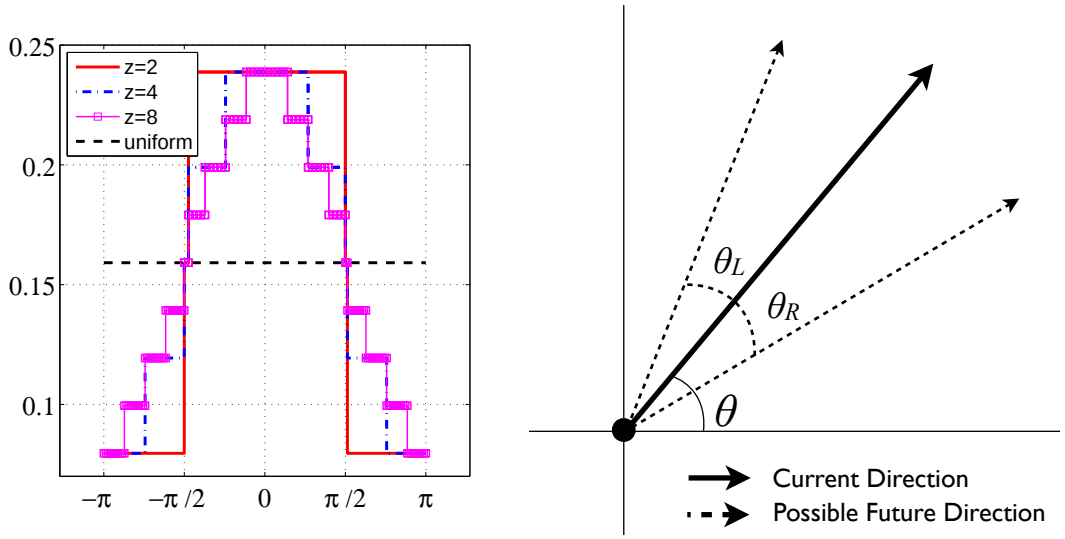
5.5.1.2 *Steady Motion based* AFP

Given a mobile user m and m's previous moving direction θ, we can compute the moving direction of m using the probability density function of moving direction θ, denoted by $p(\theta)$. $p(\theta)$ is uniformly distributed and $p(\theta)$ is $\frac{1}{2\pi}$ if the mobile client selects the next direction randomly, as shown in Figure 42(a).

However, under the steady motion assumption, the mobile client is likely to increase or decrease the value of θ but not dramatically. For example, at an intersection, the probability of making a U-turn for the mobile client is less than the probability of making a left or right turn. Therefore, the density function $p(\theta)$ is not uniformly distributed. The modified $p(\theta)$ is provided as follows:

$$p(\theta) = \begin{cases} \frac{1+\frac{y}{z}\lceil\frac{\frac{\pi}{2}-|\theta|}{\frac{y\pi}{z}}\rceil}{2\pi} & if\ \theta \in [-\frac{1}{2\pi}, \frac{1}{2\pi}] \\ \frac{1-\frac{y}{z}\lceil\frac{-\frac{\pi}{2}+|\theta|}{\frac{y\pi}{z}}\rceil}{2\pi} & otherwise \end{cases}$$

Here y and z are parameters of steadiness such that $\frac{y}{z} < 1$. Figure 42(a) shows the probability density function $p(\theta)$ for different values of z when $y = 1$.



(a) $p(\theta)$ (b) Moving Direction

Figure 42: Steady Motion Assumption

Figure 42(b) shows the steady motion behavior over the moving direction θ while θ may change between $-\theta_L$ and $-\theta_R$. Based on this assumption we define a steady motion

distance $D_{steady}(m, \text{AFR}_m)$, as follows:

$$\frac{1}{\theta_R + \theta_L} \int_{-\theta_L}^{+\theta_R} D_\theta(m, \text{AFR}_m) d\theta \tag{38}$$

where $D_\theta(m, \text{AFR}_m)$ is the distance from the current location of m to the intersection point with the boundary of AFR_m over which m may cross while heading towards the θ direction.

The steady motion based alarm free period for mobile client m, denoted by AFP_m, is computed as follows:

$$\text{AFP}_m = \frac{D_{steady}(m, \text{AFR}_m)}{V_m} \tag{39}$$

5.5.2 Extending Alarm Free Regions

We have shown that the Mondrian tree indexes both spatial alarm regions and empty regions, and thus we can efficiently determine whether a mobile user is inside of a spatial alarm region or an alarm free region. Furthermore, by utilizing alarm free region (AFR), we can significantly reduce the number of unnecessary alarm evaluations in anticipation of mobile client movement.

However, directly using empty regions as alarm free regions can incur higher number of region crossing checks to be performed, especially when mobile clients travel from one small empty region to another. Thus, the gain from reduction of the number of unnecessary alarm evaluations is offset by the cost of higher AFR crossing checks when AFRs are small in size. We below examine the cost of AFR based alarm evaluation in order to better understand the impact of the AFR perimeter on the cost of alarm evaluation.

Assume that the mobile user m moves in a randomly chosen direction with a constant speed V_m, C_{SA} is the cost for one alarm evaluation, θ is the angle between m's moving direction and the positive x-axis (as shown in Figure 42(b)), $l(\theta)$ is the distance from the current location of m to the intersection point p with the boundary of AFR_m when m travels along the θ direction, $\lambda(\text{AFR}_m)$ is the perimeter of AFR_m, and V_m is m's current speed. Given m's current alarm free region AFR_m, we can compute the amortized alarm evaluation

cost for m over time, denoted by C_m, as follows:

$$C_m = C_{SA} \times (\int_0^{2\pi} \frac{l(\theta)d\theta}{2\pi \cdot V_m})^{-1} = \frac{C_{SA} \cdot 2\pi \cdot V_m}{\lambda(\text{AFR}_m)} \quad (40)$$

Note that $\int_0^{2\pi} l(\theta)d\theta = \lambda(\text{AFR}_m)$, and given that AFR_m is a rectangle region, the intersection point p should be unique. Based on this equation, the average alarm evaluation cost is minimized when the perimeter of the AFR_m is maximized.

This motivates us to investigate the opportunities of composing larger AFRs for each mobile client by merging empty regions in the vicinity of the mobile client. Intuitively, by maximizing the perimeter of AFRs, we can minimize the number unnecessary region crossing checks to be performed, which further minimize the average cost of alarm evaluation. This is because reducing region crossing checks can lead to better energy efficiency at mobile clients and reduced communication and computation load at the server.

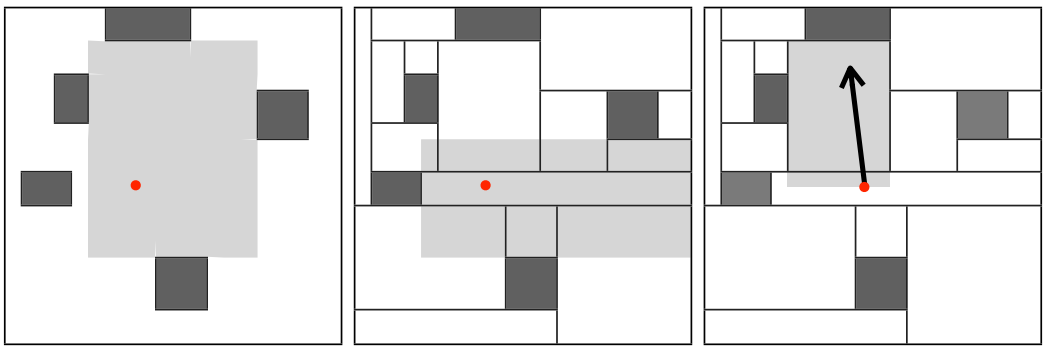
In the rest of this section, we describe two optimization techniques for extending AFR.

5.5.2.1 Patch and Trim (PAT)

Before describing our technique for merging empty regions to form a larger AFR, we illustrate the technical challenge of this problem by example. Figure 43(a) has five spatial alarms shown in small dark grey rectangles and the red circle denotes the current location of mobile client m. Clearly the optimal AFR with respect to these five spatial alarms and the current location of m is the light gray rectangle in Figure 43(a). However, it is costly in general to compute such an optimal AFR. According to [26], given a set of n spatial alarms, the time complexity for computing the largest empty rectangle with respect to the n alarms is $O(n \log^3 n)$. Therefore, in this paper we develop a near-optimal but fast algorithm to compute an extended AFR. We call it Patch and Trim (PAT).

Figure 43(b) shows the result of constructing an AFR rectangle by patching the adjacent AFRs (empty regions) and trimming the patched polygonal region over orthogonal lines.

PAT consists of two phases: *Patch* phase and *Trim* phase. In the *Patch* phase, we search the Mondrian index to get a set of adjacent AFRs with respect to the current location of mobile client m by using REGIONSEARCH in $O(\log n)$. Then we perform the *Trim* phase

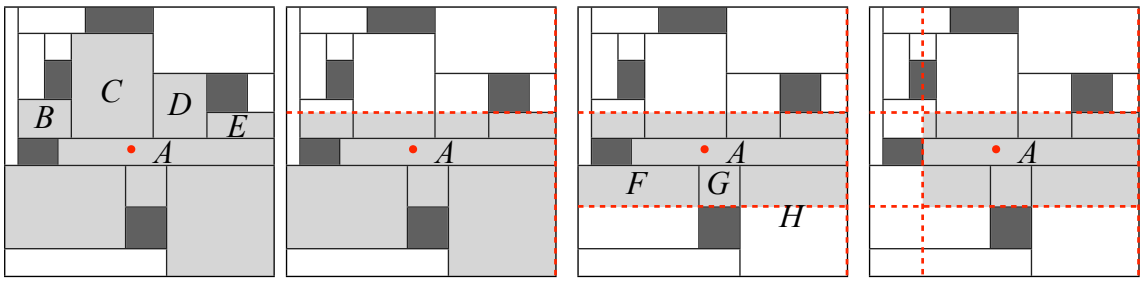


(a) Optimal (b) PAT (c) MPAT

Figure 43: Examples of extending AFRs

over the four sides (top, right, bottom, left) of the patched empty region in the clockwise manner in $O(1)$. In total, PAT takes $O(logn)$, which is faster than $O(n \log^3 n)$.

Figure 44 illustrates the patch phase and the four steps of the $Trim$ phase. Assume that the user is inside of empty region A. Figure 44(a) shows the resulting polygonal area of the patch phase where all neighboring empty regions of A are selected. In the trim phase, we need to trim the polygonal area into a rectangle region containing A by setting the boundary of the extended AFR. This is done by selecting the intersecting interval of neighboring empty regions on four sides of A, clockwise one at a time. For example, in order to extend A upward, we consider A's neighboring empty regions: B, C, D, and E. The intersecting y interval by all four rectangles is the same as E's height. Therefore we choose y value of top border in E as the extended top boundary shown in 44(b). On the right side of A, there are no adjacent AFRs. Therefore we use the x value of A's right border as the right boundary of extended AFR, shown in Figure 44(b). In order to extend A downward, we examine A's downward neighboring empty regions: F, G, and H and the intersecting y interval they share is the same as G's y interval (height). Thus we select y value of G's bottom border as the bottom boundary of the extended AFR, shown in Figure 44(c). Finally we examine if we can extend A on its left border. Given that the left neighboring region of A is a spatial alarm, we cannot extend A further on its left side. Thus, we choose the x value of A's left border as the left boundary of the extended AFR, shown in Figure 44(d).



(a) Patch (b) Top and Right (c) Down (d) Left

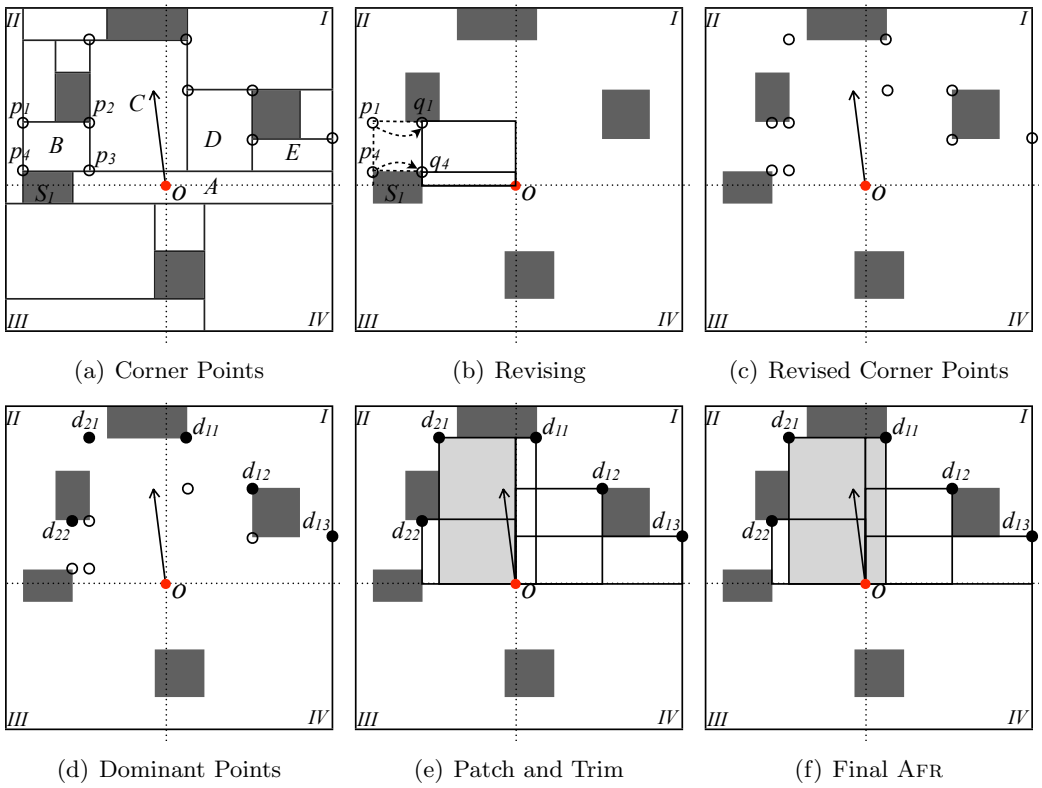
Figure 44: Patching and Trimming steps

We have shown that PAT can quickly compute an extended AFR that is near optimal when there is no additional knowledge about user's mobility and motion behavior. As discussed in Section 5.5.1, given the current AFR of a mobile user m, we compute the alarm free period (AFP) during which m can hibernate its alarm evaluation service. Obviously, the longer is an AFP, the less number of AFR crossing checks will be performed and thus less alarm evaluations for m.

Given that most of mobile users travel on a spatially constrained road network or walk path with a destination in mind, we know the approximate travel direction for these mobile users. Under such steady motion assumption, we can utilize the moving direction of a mobile user to compute the extended AFR such that the alarm free period (AFP) computed using this extended AFR will be maximized.

Recall the example in Figure 43, if user m is moving northwest, then all spatial alarms located in the south of m's current location are no longer relevant. Thus, the best extended AFR for m in this case is the extended AFR computed using MPAT, shown in Figure 43(c). This is because by utilizing the steady motion of m, the extended AFR in Figure 43(c) maximizes the AFP for m.

This motivates us to develop a motion-aware patch and trim algorithm that can compute the extended AFR for each mobile user m based on her motion behavior, aiming at maximizing the AFP for m.



(a) Corner Points (b) Revising (c) Revised Corner Points

(d) Dominant Points (e) Patch and Trim (f) Final AFR

Figure 45: Computing a Motion-aware AFR

5.5.2.2 Motion-Aware Patch and Trim (MPAT)

The motion-aware patch and trim algorithm for extending AFR consists of five steps. Due to space limit, we omit the pseudo code in this paper and provide a walk-through of the algorithm using the example in Figure 45.

Step 1: Determine the relevant Quadrants. Under the steady motion assumption (recall section 5.5.1), if a mobile user is heading towards the θ direction, then the probability of m moving forward or turning left or right on its current location is much higher than the probability of making a U-turn, as shown in Figure 42(a). Therefore, spatial alarms in the downward direction of m are no longer relevant.

In order to make the best use of the steady motion behavior of the mobile user m, we partition the universe of discourse into four quadrants using the current location of m as the center such that only those quadrants that are relevant to computing the extended AFR of m will be selected.

Figure 45(a) shows an example of four quadrants partitioned at o, the current location of mobile user m. We determine the set Q of quadrants to be considered based on the steady motion density function $p(\theta)$ of the mobile user m (recall Figure 42(a)). $|Q|$ is at most 2. If θ is 100° and we use $p(\theta)$ with $z = 8$, then the range of moving direction will be between 78.5° and 122.5 °. The range of direction overlaps with Quadrant I and II. Therefore we consider two quadrants out of four. If the range of moving direction overlaps with only one quadrant (e.g., when θ is 30°), then only one quadrant is relevant and selected in this step.

Step 2: Find Candidate Corner Points. Let A be the empty region in which m resides. In each selected quadrant, we first use the the function RegionSearch to find all neighboring empty regions of A, which are included in this quadrant. Then we examine each of neighboring empty regions, and add its four corner points into the set of candidate corner points. This set of candidate corner points will be used to determine the component rectangle of the extended Afr in the selected quadrant. In our running example, corner points in the quadrants I and II are represented by hollow circles as shown in Figure 45(a).

Step 3: Find candidate component rectangle by revising corner points. For each selected quadrant, we examine all candidate corner points and revise those that may not form a component rectangle of the extended Afr. A component rectangle of the extended Afr is the empty rectangle region that has o, the current position of m, as one of the corner points. For example, we examine the four corner points of B in the left top quadrant as shown in Figure 45(a). Consider a component rectangle consists of p_1 and o. This component rectangle overlaps with the spatial alarm S_1 as shown in Figure 45(b). Therefore the corner point p_1 needs to be moved to q_1 so that it avoids to overlap with S_1. Similary, a corner point p_4 should be revised to move to q_4 which has the same x-value as S_1 's right border. The new component rectangle formed with o and q_2 as two diagonal corner points will not overlap with any spatial alarms. The same process runs iteratively until every candidate corner point is examined and revised. The revised corner points are shown by dashed arrows in Figure 45(c).

Step 4: Find Dominating Points. Let Quadrant I denote the upper right quadrant,

Quadrant II denote the upper left quadrant, Quadrant III denotes the bottom left quadrant, and Quadrant IV denotes the bottom right quadrant. In each quadrant we change the meaning of the dominant points for $p_1(x_1, y_1)$ and $p_2(x_2, y_2)$ as follows:

(a) Quadrant I: p_1 dominates p_2 if $x_1 \geq x_2$ and $y_1 \geq y_2$

(b) Quadrant II: p_1 dominates p_2 if $x_1 \leq x_2$ and $y_1 \geq y_2$

(c) Quadrant III: p_1 dominates p_2 if $x_1 \leq x_2$ and $y_1 \leq y_2$

(d) Quadrant IV: p_1 dominates p_2 if $x_1 \geq x_2$ and $y_1 \leq y_2$.

Based on the definition above, the set of dominating points are represented as solid black dots in Figure 45(d), namely d_{21}, which disregards two hollow points with the same x values, d_{22}, which disregards three hollow points, d_{11}, which disregards one hollow point with the same x-value, d_{12}, which disregards two hollow points, one with the same x-value and the other with the same y-value; and d_{13}.

All the hollow dots are dominated by the black dot corner points. An important property of a dominating point is that the size of the component rectangle defined by the current location o of m and a given black dot corner point is maximized, compared to the component rectangle defined by the current location o of m and a hollow corner point dominated by the given black dot. For example, the component rectangle with a dominant black dot is larger than the component rectangle with a hollow corner point.

Step 5: Patch and Trim Component Rectangles. The set of dominating points form corners of component rectangles in each quadrant. The final AFR is composed by patching one component rectangle from each quadrant and trimming the patched rectangle so that the distance from the current location to the border of resulting AFR rectangle is maximized while m heading towards the θ direction. In Figure 45(e), there are five component rectangles, each corresponds to one of the five dominating points marked in black dot. In quadrant II, we select the component rectangle with o and d_{21} instead of the one with o and d_{22} because the former provides the longest distance to the border. Similarly, in quadrant I we choose the component rectangle with o and d_{11}, because the

distance to the border remains the longest. If we choose d_{12}, then the final AFR is wider but shorter and the distance to the border is shorter.

The time complexity for computing motion-aware AFR is $O(n^2)$, given that Step 1 finishes in $O(1)$, step 2 in $O(n \log n)$, step 3 in $O(n^2)$, step 4 in $O(n \log n)$ by divide and conquer, and step 5 takes $O(n)$.

5.5.3 Distributed Mondrian Index

Given n mobile users subscribing to public and private spatial alarms, there are three alternative ways of creating and maintaining Mondrian indexes. First, we can create and maintain a single Mondrian tree for all mobile users and all their spatial alarms. We call it the *centralized* approach. Alternatively, we can create n Mondrian tree indexes, each is devoted for one mobile subscriber, which indexes all alarms subscribed by this subscriber, including public, private, and shared alarms. We call it the *distributed* approach. The third alternative is to create one Mondrian index for all public alarms, and n Mondrian tree indexes for all private and shared alarms, each is dedicated to one of the n mobile subscribers. We call it the *Hybrid* approach.

Given the specific characteristics of spatial alarms, the approaches of creating and maintaining individual Mondrian trees, one per client, can significantly minimize the overhead of searching for relevant alarms and AFRs of a given mobile user.

For example, if Alice installed 10 spatial alarms and Bob installed 30 alarms, then we create a single Mondrian tree with 40 alarms under the centralized approach and two Mondrian trees, one for Alice with 10 alarms and the other for Bob with 30 alarms, under the distributed approach. Then, the average size of AFRs in the centralized approach will be much smaller than that of AFRs in the distributed approach because the centralized approach inserts more alarms in a Mondrian tree. Furthermore, the less nodes we have in a Mondrian tree, the faster it takes to find a leaf node because the search conducted by Alice will not be affected by the spatial alarms installed by another user Bob. In addition, three alternative system architectures can be used to support spatial alarm processing: server-centric, client-centric, and distributed client-server. Considering that the client-centric architecture

is only applicable for processing private spatial alarms [72], we below focus on server-centric architecture and distributed client-server architecture.

In the server-centric architecture, spatial alarms will be installed, subscribed and processed at the server and mobile clients do not contribute directly to the spatial alarm processing tasks. Mobile clients only receive alarm notification when entering their alarm target regions.

In the distributed client-server architecture, the server creates and maintains one Mondrian index per mobile subscriber. Insertion of new public or shared alarms will trigger the server to broadcast the newly installed alarms to those mobile clients whose subscriptions match with this new alarms, so that each mobile client can insert the newly added alarm into its local Mondrian tree. Insertion of a private alarm only involves the insertion of this alarm to the local Mondrian tree of its owner. A spatial alarm is removed from the system (client and server) upon reaching its expiration time. Alarm expiration is checked at each alarm evaluation. Spatial alarm processing is accomplished by the server and the mobile clients collectively. Several strategies can be used for partitioning of spatial alarm processing tasks into server side and client side processing. We below describe three possible strategies for implementing the Mondrian tree approach under the distributed client-server architecture.

The first strategy will have the sever perform the following four tasks: (1) construct and maintain the Mondrian tree for each mobile object m; (2) search the Mondrian tree index to find AFR_m for m; and (3) compute AFP_m for m; and (4) send AFP_m to m. In this scenario, the client application checks if AFP_m expires. If so, it sends a new AFP_m request message to the server.

The second strategy will have the server perform only the first two tasks and a modified version of the 4th task. Concretely, the server sends the current AFR_m to m instead of the AFP_m. Now the client computes AFP_m locally using AFR_m (task 3). The client only reports to the server when it moves outside of AFR_m or an alarm monitoring region.

The third strategy will have each client build a Mondrian tree. Each client lookups the index locally, finds its current AFR, and computes AFP accordingly. A client only reports

to the server if it arrives at an alarm monitoring region. All public, private and shared alarms are installed at the server and distributed to clients by the server.

The choice of which strategy to use depends primarily on the capacity of mobile clients. Some clients have limited capability in terms of battery power, computing and storage capacity, whereas others are equipped with computing and memory capability equivalent to a laptop computer such that it can store the Mondrian tree that indexes all of its subscribed alarms locally. In this situation, a mobile subscriber who is capable of storing its own Mondrian tree locally can also perform the Mondrian tree lookup and AFP computation locally. Thus this strategy significantly reduces the amount of client to server communication cost.

5.6 Experiments and Evaluation

In this section we report our experimental evaluation of the performance and effectiveness of the Mondrian tree approach to spatial alarm processing. Our experiments are conducted with two objectives. First, we want to compare Mondrian index with a set of popular spatial indexing techniques, such as R-tree, k-d tree, and Quadtree in terms of multiple parameters, including total time, index lookup time, memory size, AFR computation time, average size of AFR, and average client-to-server communication messages. Our experimental results show that Mondrian tree approach significantly outperforms R-tree, k-d tree, and Quadtree for spatial alarm processing, due to faster AFR computation, longer AFP, and indexing of both alarms and AFRs with Mondrian trees. Second, we conduct experiments to compare different design choices for processing spatial alarms using Mondrian tree indexes, including (i) comparing the basis AFP (alarm free period) and the steady motion based AFP with respect to periodic evaluation; (ii) comparing the basic Mondrian tree with Mondrian-PAT and Mondrian-MPAT in terms of the number of AFR crossing checks required and average AFP interval length; (iii) comparing centralized, distributed, and hybrid approach to creating and maintaining Mondrian indexes in terms of AFR computation and average size of AFR.

5.6.1 Experiment Setup

We extend GTMobiSim mobility simulator [76] and generate a set of traces of moving vehicles on a real world road network. The road network data are obtained from the National Mapping Division of the U.S. Geological Survey[13] in the form of Spatial Data Transfer Format[9]. Vehicles are distributed randomly on the road segments according to traffic densities determined from the traffic volume data [40]. These vehicles move on the roads of metro Atlanta. The number of spatial alarms and the number of vehicles vary from 1,000 to 10,000. The default number of vehicles is 5,000 and the default number of alarms is 5,000.

For R-tree, Quadtree, and k-d tree, the index structure does not provide a good quality of AFR. In this experiments we compute AFRs by the safe region algorithm in [18] for distributed processing of spatial alarms. It consists of two steps. First, it finds n nearest alarms and computes the AFR using a greedy approach in $O(n^2)$. The parameter n is a system supplied parameter. When n is too small, AFR is too small. On the other hand, if n is too big, the AFR computation is too expensive.

All the Mondrian trees used in this experimental evaluation are centralized Mondrian tree except in the experiment for performance of multiple Mondrian trees. Given that Mondrian tree is a memory based space partitioning index structure, we modified other data structures so that they stored data in the main memory.

All experiments run on a Linux machine with Intel Core 2 Duo CPU 2.8GHz and 4GB RAM.

5.6.2 Effectiveness of Mondrian Tree Index

The first set of experiments is designed to compare the Mondrian tree against other data structures in spatial alarm processing by measuring total time, index lookup time, memory size, AFR computation time, average size of AFR, and average client-to-server communication messages. In order to understand how the number of alarms and the number of users impact the alarm process performance, we design two groups of experiments, one with varying number of users from 1,000 to 10,000 and 5,000 alarms (Figures 46(a) - (d)) and the

other with varying the number of alarms from 1,000 to 10,000 and 5,000 vehicles (Figures 46 (e) (h)). Note that to be fair in the comparison of centralized Mondrian tree with other index structures, all experiments reported in this first set is using the basic Mondrian tree without AFR or AFP optimizations.

Figures 46(a) and 46(e) shows total time spent for alarm processing for varying number of alarms and varying number of users respectively. In both figures, although the total time increases for all approaches as the number of alarms or users increases, the rate of increase for Mondrian tree is significantly smaller comparing with R-tree and k-d tree. The reason Quadtree has slightly slower than Mondrian tree is that compared to other two data structures it only concerns a small leaf node and a set of spatial alarms in it, which reduces the number of alarms to be considered for computing AFRs.

Figures 46(b) and 46(f) shows total AFR computation time. By comparing with Figures 46(a) and 46(e), we note that AFR computation takes up to 90% of total alarm processing time. On each AFR request by mobile user m, the server looks up spatial alarms with regard to m's current location. Given a location, Mondrian tree approach does not compute AFRs. It just looks up the relevant leaf node. However, R-tree and k-d tree need to dynamically compute AFRs upon each user request in $O(n^2)$, so their AFR computation costs are quadratic in comparison with Mondrian approach.

Figures 46(c) and 46(g) compares the average size of AFRs. The Mondrian tree partitions the universe of discourse into relatively smaller size of rectangles on average and thus smaller AFRs. Therefore the average size of AFRs in the Mondrian tree is smaller than one in R-tree or k-d tree as shown in 46(c). Like Mondrian tree, Quadtree is also a region partition tree. Therefore the average size of AFR is also small. As we increase the number of alarms, the average size of AFRs decreases because more empty regions are occupied by alarms. However, it is obvious that the average size of AFRs does not change much while fixing the number of alarms and varying the number of users as shown in Figure 46(g).

The message cost from client to server is shown in Figures 46(d) and 46(h). Due to the smaller size of AFR in Mondrian tree, the average number of client-to-server message for requesting AFRs is slightly larger than other data structures. Quadtree also indexes

smaller empty regions, but once it enters a region that has a spatial alarm, then it needs to compute AFR. This makes users using Quadtree frequently cross the border and compute AFRs. Therefore it has the biggest number of message cost as shown in Figure 46(d). When we vary the number of mobile objects but fix the number of alarms in Figure 46(h), the number of messages requesting AFRs barely changes because average size of AFRs are similar for all mobile objects, and thus each has similar probability of crossing AFRs.

5.6.3 Optimization Techniques

The second set of experiments compared optimization techniques such as AFP approaches and extending AFRs. Figure 47(a) shows that the basic AFP and the steady-motion AFP have much longer AFP interval compared to periodic approach and thus reducing the amount of unnecessary alarm evaluations. Also steady-motion AFP outperforms the basic AFP though the difference is decreasing as the number of alarms increases, because more alarms implies smaller AFRs. Figure 47(b) shows that the average amount of time for computing AFP is less than $1ms$, compared to the average AFP interval of 8 minutes and 30 seconds for 10,000 alarms. The experiment is the simulation of 30-minute driving. If a user actually moves for 30 minutes as the simulation, the spatial alarm processing can enter hibernation for about 8 minutes and 30 seconds, 28% of 30 minutes of driving. Also Figure 47(b) shows that although steady motion AFP has the longest AFP interval, it takes more time to compute AFP because it needs to compute the average of AFPs for more than one direction.

Figures 47(c) and 47(d) show the comparison of the basic AFR with the two AFR optimizations. In this set of experiments, **Mondrian** refers to the use of empty region from the basic Mondrian tree construction as AFRs (no optimization). **Mondrian-MPAT** refers to Mondrian approach powered with MPAT, and **Mondrian-PAT** is the Mondrian approach powered by PAT. In all cases, we compute AFP based on AFRs for processing spatial alarms. As shown in Figure 47(c), the message cost of **Mondrian-MPAT** is smaller than **Mondrian-PAT** because **Mondrian-MPAT** considers moving direction and extends an AFR along the moving direction so that users may cross the border less frequently and also ask a new AFR less frequently than other approaches. Furthermore, extended AFRs based

on **Mondrian-MPAT** also provide longer AFP as shown in Figure 47(d) because it has a longer distance from the current location to the border of AFR.

5.6.4 Performance of Distributed Mondrian Trees

In this set of experiments, we compare the performance of distributed Mondrian trees against the performance of centralized Mondrian index and hybrid Mondrian index. We set the number of users to be 5,000, the number of public alarms to be 100, and increase the number of private alarms per user from 0 to 10. Therefore the total number of alarms vary from $100, (5,000 + 100), (10,000 + 100), \cdots, (50,000 + 100)$.

Figure 48(a) shows that **Centralized Mondrian tree** takes the longest time for computing AFR. If we add one additional private alarm to for all users, then **Centralized** Mondrian index needs to add 5,000 (1 alarm $\times$ 5,000 users) more alarms in the tree. On the other hand, the **Hybrid** and **Distributed** Mondrian indexing increase only 1 alarm. Similarly, when we add 10 additional private alarms per user, then **the Centralized Mondrian** will add 50,000 (10 alarms $\times$ 5,000 users) more alarms. Therefore **Centralized Mondrian** has the worst performance. When there are 100 percent of public alarms, then there is no difference in each approach. **Hybrid** needs to check two trees: one for public alarms and the other for private alarms. Hence it takes more time than **Distributed**.

Figure 48(b) shows that **Centralized Mondrian approach** has the smallest average AFR size. On 100 percent public alarms, the size of AFR is the same because each approach has the same alarms. When we assign one additional private alarms to each user, then **Centralized Mondrian** will have 5,000 more alarms while distributed or Hybrid Mondrian approaches will an increase by only 1 alarm. Therefore as we increase the number of private alarms per user, the average size of AFR in the **centalized** Mondrian tree decreases dramatically.

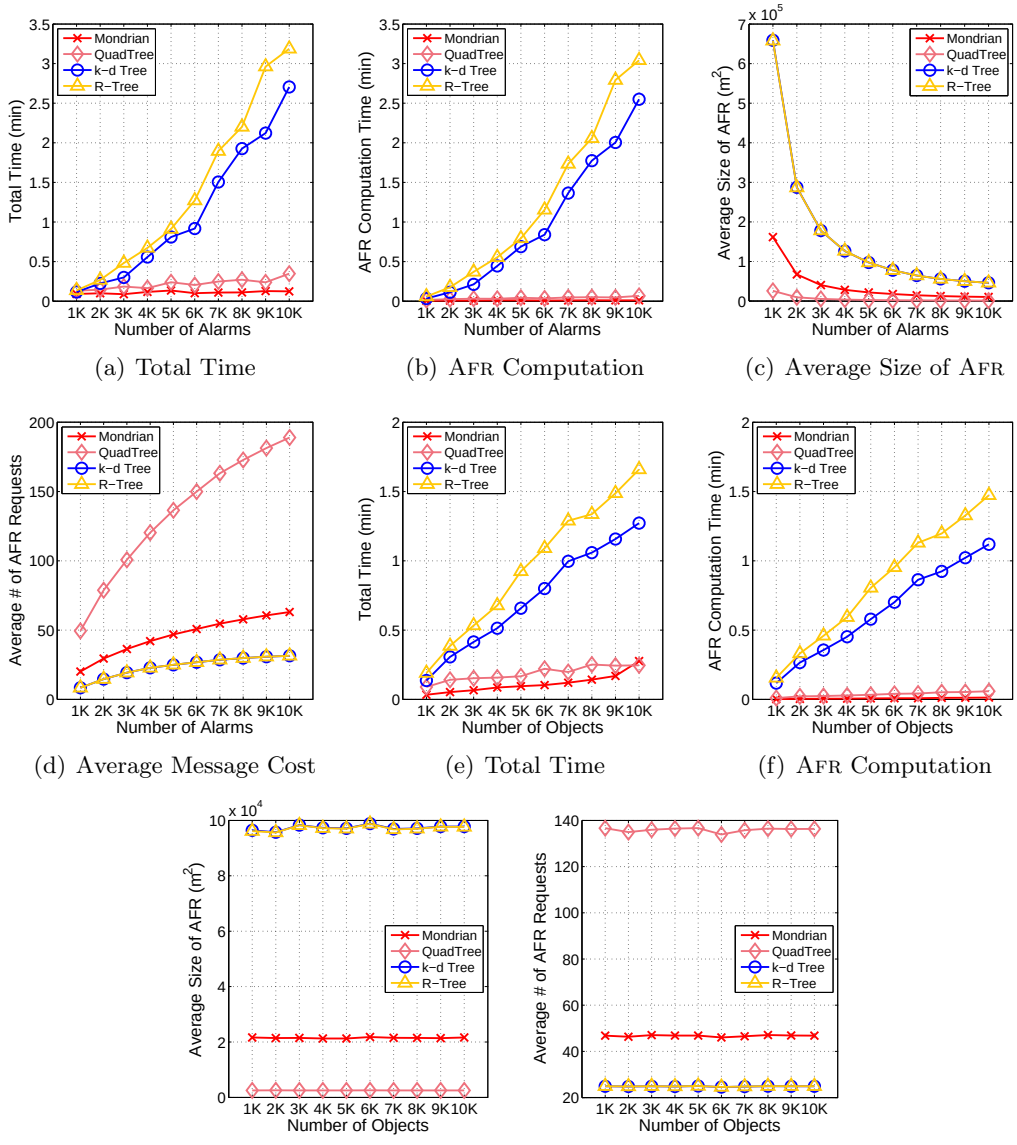
5.6.5 Effectiveness of Mondrian*

We presented Mondrian*, an optimized Mondrian tree, in Section 5.4. We performed a set of experiments to measure the performance of Mondrian* against Mondrian basic in terms of memory size and average depth of the tree. Figure 48(c) shows the average depth

of Mondrian and Mondrain* by varying the number of spatial alarms from 1K to 10K. Clearly, Mondrian* has the shorter depth on average for a couple of reasons. First, the number of children nodes that a non-leaf node may have in Mondrian* tree is four, whereas it is two in the Mondrian tree. Second, in batch construction of Mondrian* tree, it chooses an alarm that is approximately located in the center of all alarms as the next alarm to be inserted so that we can distribute remaining alarms into the four children evenly. Figure 48(d) shows the memory size. Recall Figure 38(h), Mondrian tree needs 9 nodes for one spatial alarm as shown in Figure 38(h) while Mondrian* tree only needs five as shown in Figure 40(b). Hence, Mondrian* tree takes shorter time to lookup and needs less memory.

5.7 Conclusion

In this chapter, we have presented the design and implementation of the Mondrian tree index, a fast index structure for scalable processing of spatial alarms. Compared with conventional spatial indexes, such as R-tree, Quadtree, and k-d tree, the main distinguishing feature of Mondrian tree, is that the Mondrian tree approach indexes not only spatial alarms but also empty regions, which enables us to look up AFRs fast compared to other data structures. Another novelty of the Mondrian tree index is its ability to utilize the characteristics of spatial alarms to create and maintain one Mondrian tree for each mobile subscriber, which is particularly effective when there is relatively small number of public alarms compared to the total number of private alarms in the system. We also provide a set of optimization techniques for scaling spatial alarm processing based on Mondrian index, such as motion-aware AFP extended AFRs using PAT and MPAT. Our experiments show that Mondrian tree can dramatically minimize the amount of unnecessary AFR and scale the spatial alarm processing, compared to R-tree, Quadtree, and k-d tree.

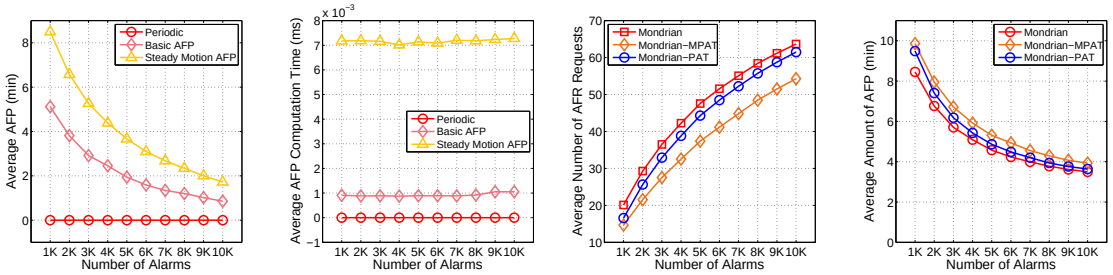


(a) Total Time (b) AFR Computation (c) Average Size of AFR

(d) Average Message Cost (e) Total Time (f) AFR Computation

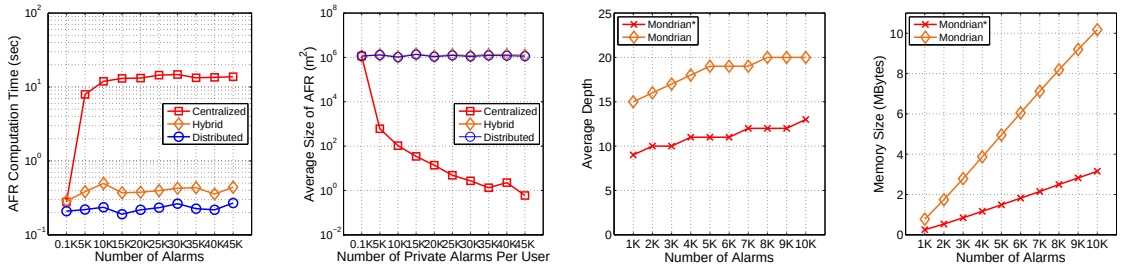
(g) Average Size of AFR (h) Average Message Cost

Figure 46: Performance Results in Server-side



(a) Average AFP (b) AFP Computation Time (c) Average Message Cost (d) Average Length of AFP

Figure 47: Optimization Techniques



(a) AFR Computation (b) Average Size of AFR (c) Average Depth (d) Memory Size

Figure 48: Server-centric vs. Distributed Approach and Mondrian vs. Mondrian*

CHAPTER VI

CONCLUSION AND OPEN ISSUES

Information diffuses over social network and spatial domain. Without reading newspaper website, people can get information from the wall of friends Facebook. Many media websites provide a link for sharing the article via SNS so that more people would read this news article through SNS. For commercial organizations, even non-commercial organizations, that want to promote a new product or diffuse new information, should consider people who have high social influence so that their new information can be delivered to as many people as possible.

Due to the advance in computing power of mobile devices and mobile communication, information diffusion in spatial domain is also important. Information on a specific location can be diffused to people who visit the location. This enables us to deliver information related to that location to people who want the information related to the location.

Despite advances in technology, a number of issues need to be considered for information diffusion. Social influence model over social networks should be built. It should consider not only the topology of SNS but also other attributes in SNS. Social influence model has a set of parameter that can be tuned to set the appropriate characteristics of SNS. Means of information diffusion in spatial domain should be considered. The ability of handling a large number of location data should be developed.

This dissertation provided solutions to these important challenges for information diffusion in social network and spatial domain. We presented (1) activity-based social influence model over social network using heat diffusion equation, (2) probability and incentive based social influence model, (3) spatial alarms as a means of distributing information in the context of location, and (4) Mondrian tree, an efficient indexing structure and a set of algorithm for scalable processing of spatial alarms.

6.1 Dissertation Conclusion

We summarize below the major contributions of this thesis towards addressing the challenges in information diffusion.

- We have presented the activity-base social influence model based on activity enhanced heat diffusion kernel and a suite of activity influence rank based top k algorithms. Our ActivityInfluence model has made three unique contributions. First, we introduce a novel mechanism to extend the heat diffusion model by effectively incorporating both interactive and non-interactive activities. Second, we develop a suite of top k influence rank based node selection algorithms by minimizing the overlapping in the node coverage of top k most influential nodes, including independent influence rank, (b) locally optimal influence rank and (c) globally optimal influence rank using Hill Climbing algorithm. Finally we conduct an extensive series of experiments on three representative real-world social network datasets to show the effectiveness of our activity-based social influence model and influence rank algorithms. Compare to the existing topology-based influence diffusion model, the activity-based social influence model considers not only topology of a social network but also activity sensitive attributes such as interactive activities, non-interactive activities and time stamps. Our ongoing research continues along two dimensions. First, we are interested in further investigating other types of social network attributes that are critical to social network analysis, such as time stamps and similarity of user profiles. Second, we are also interested in studying different types of incentives and how they impact on the social influence and the spread of information over a given social network.

- We have presented the social influence model through probability and incentives based on enhanced activity information. Compared to previous heat diffusion and topology-based probability models, our model contributes in three aspects. First, in order to express the real world more accurately, we introduce various system parameters. These parameters such as activeness $A(u)$, probability to be a stopper, and θ_I to filter out acquaintances. Second, we develop a incentive model so that any types of

incentives can be used once it is normalized as a number between 0 and 1. By this incentive models, we mimic the boosting effect that the real incentives actually have. Reward effects boost activeness $A(u)$ so that a node u may have lower chance to be a stopper and higher chance to activate her friends. Finally we conduct an extensive series of experiments on various parameters and performance of our models. From the experimental results, we show that by tuning the parameter we can precisely mimic the real world. Our probability and incentive approach maximizes propagating innovations which can be a new idea or new product over the given network.

- We make two important contributions towards supporting spatial alarm based mobile applications [18]. First, we introduce the concept of safe period to minimize the number of unnecessary alarm evaluations, increasing the throughput and scalability of the system . Second, we develop a suite of spatial alarm grouping techniques based on spatial locality of the alarms and motion behavior of the mobile users, which reduces the safe period computation cost for spatial alarm evaluation at the server side. We evaluate the scalability and accuracy of our approach using a road network simulator and show that the proposed safe period-based approach to spatial alarm processing offers significant performance enhancements for alarm processing on server side while maintaining high accuracy of spatial alarms.

- We developed the design and implementation of the Mondrian tree index, a fast index structure for scalable processing of spatial alarms [36, 34, 35, 65]. Compared with conventional spatial indexes, such as R-tree, Quadtree, and k-d tree, the main distinguishing feature of Mondrian tree, is that the Mondrian tree approach indexes not only spatial alarms but also empty regions, which enables us to look up AFRs fast compared to other data structures. Another novelty of the Mondrian tree index is its ability to utilize the characteristics of spatial alarms to create and maintain one Mondrian tree for each mobile subscriber, which is particularly effective when there is relatively small number of public alarms compared to the total number of private alarms in the system. We also provide a set of optimization techniques for

scaling spatial alarm processing based on Mondrian index, such as motion-aware AFP extended AFRs using PAT and MPAT. Our experiments show that Mondrian tree can dramatically minimize the amount of unnecessary AFR and scale the spatial alarm processing, compared to R-tree, Quadtree, and k-d tree.

6.2 Open Issues and Future Research Directions

Although this dissertation work has tried to address as many as issues for development of information diffusion for social network and spatial domain, but there are still open questions. We discuss this set of open issues in this section.

Social Influence and Privacy Social influence computation may require various of user data; age, gender, location, education, marital status, work history, comments, photos, videos, time stamps of actions, timestamps of friendship established, etc. The problem is that these information are private and difficult to collect without permissions from users. With the supply of more attributes in SNS without violating privacy, we hope to continue enhance our incentive model.

Spatial Alarm Processing Algorithms for Moving Objects: In this dissertation, we assume that alarms are static and only alarm subscribers are moving. There are other cases such as (a) moving alarms with static subscribers and (b) moving alarms with moving subscribers. For example, in a cold day morning, parents set an alarm on a school bus. When a school is near a half mile away, then parents take their children out for the school bus. Another example is that a client X sets an alarm on her friend Y. When Y enters a specific build Z, then the alarm will be triggered. Currently Mondrian tree is not able to index moving objects. Future extension of spatial alarm processing requires Mondrian tree to index moving objects so that we can handle both (a) and (b) cases.

Spatial Alarm Processing Algorithms over Cloud Infrastructure We discussed a suite of distributed alarm processing algorithm based on the computing power of mobile devices. Some device may store all of alarms installed in the server, while others may only provide location information without storing any alarms. Currently we assume that there is only one server. The way of distributed approach is that clients will be responsible for some

part of computing alarms free regions or alarm evaluation. Computing power of mobile device determines how much computing should be handled in the mobile device. With the current trend in information processing moving towards cloud-based approach, spatial alarm processing also can be moved to cloud-based environment.

REFERENCES

[1] "Epinions." http://www.epinions.com.

[2] "Facebook." http://www.facebook.com.

[3] "Foursquare." http://www.foursquare.com/.

[4] "Geominder." http://ludimate.com/products/geominder/.

[5] "Klout." http://www.klout.com.

[6] "LinkedIn." http://www.linkedin.com.

[7] "Naggie 2.0: Revolutionize reminders with location!." http://www.naggie.com/.

[8] "PeerIndex." http://www.peerindex.com.

[9] "Spatial Data Transfer Format." http://www.mcmcweb.er.usgs.gov/sdts/.

[10] "The DBLP Computer Science Bibliography."

[11] "Twitter." http://www.twitter.com.

[12] "Twitter on track for 500 million total users by march, 250 million active users by end of 2012." http://www.mediabistro.com/alltwitter/twitter-active-total-users-b17655.

[13] "US Geological Survey." http://www.usgs.gov/.

[14] A. Chevalier, J. and Mayzlin, D., "The effect of word of mouth on sales: Online book reviews," *NBER Working Paper Series*, 2005.

[15] Anger, I. and Kittl, C., "Measuring Influence on Twitter," in *Proceedings of International Conference on Knowledge Management and Knowledge Technologies*, 2011.

[16] Aurenhammer, F., "Voronoi diagrams - a survey of a fundamental geometric data structure," *ACM Computing Surveys*, 1991.

[17] Bamba, B., Liu, L., and Yu, P. S., "Scalable processing of spatial alarms," tech. rep., Georgia Institute of Technology, 2008.

[18] Bamba, B., Liu, L., Yu, P. S., Zhang, G., and Doo, M., "Scalable Processing of Spatial Alarms," in *Proc. High Performance Computing*, 2008.

[19] Bao, H. and Chang, E. Y., "AdHeat: an influence-based diffusion model for propagating hints to match ads," in *Proceedings of WWW*, 2010.

[20] Bass, F. B., "A new product growth for model consumer durables," *Management Science*, 1969.

[21] Bentley, J. L., "Multidimensional binary search trees used for associative searching," in *SIGMOD*, 1975.

[22] Berger, E., "Dynamic Monopolies of Constant Size," *Journal of Combinatorial Theory Series*, 2001.

[23] Boyd, D. and Heer, J., "Profiles as Conversation: Networked Identity Performance on Friendster," in *Proceedings of HICSS*, 2006.

[24] Buskens, V. and Yamaguchi, K., "A new model for information diffusion in heterogeneous social networks," *Sociological Methodology*, vol. 29, no. 1, pp. 281–325, 1999.

[25] Cha, M., Haddadi, H., Benevenuto, F., and Gummadi, K., "Measuring user influence in twitter: The million fallacy," in *Proc. AAAI*, 2010.

[26] Chazelle, B., Drysdale, R., and Lee, D., "Computing the largest empty rectangle," in *STACS 84*, vol. 166, Springer Berlin / Heidelberg, 1984.

[27] Chen, W., Wang, C., and Wang, Y., "Scalable influence maximization for prevalent viral marketing in large-scale social network," in *Proc. KDD*, 2010.

[28] Cheng, A., "Six degree of separation, twitter style," 2010. http://www.sysomos.com/insidetwitter/sixdegrees/.

[29] Chongfu, H., "Principle of information diffusion," *Fuzzy Sets and Systems*, vol. 91, no. 1, pp. 69 – 90, 1997.

[30] Chongfu, H., "Principle of information diffusion," *Fuzzy Sets and Systems*, 1997.

[31] Dey, A. K. and Abowd, G. D., "Cybreminder: A context-aware system for supporting reminders," in *Proc. International Symposium on Handheld and Ubiquitous Computing*, 2000.

[32] Dolde, W. and Tirtiroglu, D., "Temporal and spatial information diffusion in real estate price changes and variances," *Real Estate Economics*, vol. 25, no. 4, pp. 539–565, 1997.

[33] Domingos, P. and Richardson, M., "Mining the network value of customers," in *Proceedings of SIGKDD*, 2001.

[34] Doo, M. and Liu, L., "Spatial Alarm Processing and Algorithms," in *Technical Report, Georgia Institute of Technology*, 2011.

[35] Doo, M., Liu, L., Narasimha, N., , and Vasudevan, V., "Efficient indexing structure for scalable processing of spatial alarms," in *GIS*, 2010.

[36] Doo, M., Liu, L., Narasimhan, N., and Vasudevan, V., "Mondrian Tree: Efficient Indexing Structure for Scalable Spatial Triggers Processing over Mobile Environment," tech. rep., Georgia Institute of Technology, 2010.

[37] Finkel, R. A. and Bentley, J. L., "Quad trees a data structure for retrieval on composite keys," *Acta Informatica*, 1974.

[38] Fraser, B. P. and Brown, W. J., "Media, Celebrities, and Social Influence: Identification With Elvis Presley," *Mass Communication and Society*, 2002.

[39] Gedik, B. and Liu, L., "MobiEyes: Distributed Processing of Continuous Moving Queries on Moving Objects in a Mobile System," in *LNCS*, 2004.

[40] Gedik, B. and Liu, L., "Location Privacy in Mobile Systems: A Personalized Anonymization Model," in *Proc. IEEE ICDCS*, 2005.

[41] Gladwell, M., "The tipping point: How Little Things Can Make a Big Difference," *Little Brown*, 2000.

[42] Goldenberg, J., Libai, B., and Muller, E., "Talk of the network: A complex systems look at the underlying process of word-of-mouth," *Marketing Letters*, 2001.

[43] Goldenberg, J., Libai, B., and Muller, E., "Using Complex Systems Analysis to Advance Marketing Theory Development," *Academy of Marketing Science Review*, 2001.

[44] Granovetter, M., "Threshold models of collective behavior," *American Journal of Sociology*, 1978.

[45] Grove, J. V., "Morbile marketer." http://mashable.com/2010/08/18/shopalerts, 2010.

[46] Gruhl, D., Guha, R., Liben-Nowell, D., and Tomkins, A., "Information diffusion through blogspace," in *Proceedings of the 13th international conference on World Wide Web*, WWW '04, (New York, NY, USA), pp. 491–501, ACM, 2004.

[47] Gruteser, M. and Grunwald, D., "Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking," in *Proc. MobiSys*, 2003.

[48] Guttman, A., "R-trees: A dynamic index structure for spatial searching," in *SIGMOD*, 1984.

[49] Hägerstrand, T., "Aspects of the spatial structure of social communication and the diffusion of information," *Papers in Regional Science*, 1966.

[50] Hasan, M., Cheema, M., Lin, X., and Zhang, Y., "Efficient construction of safe regions for moving $k$nn queries over dynamic datasets," in *Advances in Spatial and Temporal Databases*, Springer, 2009.

[51] Henrich, A., Six, H.-W., and Widmayer, P., "The lsd tree: spatial access to multidimensional point and non-point objects," in *Proc. VLDB*, 1989.

[52] Ho, J. Y. C. and Dempsey, M., "Viral marketing: Motivations to forward online content," *Journal of Business Research*, 2010.

[53] Iribarren, J. L. and Moro, E., "Impact of human activity patterns on the dynamics of information diffusion," *Phys. Rev. Lett.*, vol. 103, p. 038702, Jul 2009.

[54] J Brown, P. R., "Social ties and word-of-mouth referral behavior," *Journal of Consumer Research*, 1987.

[55] J Coleman, H. M. E. K., *Medical Innovations: A Diffusion Study.* Bobbs Merrill, 1966.

[56] Kelman, H., "Compliance, identification, and internalization: Three processes of attitude change," *Journal of Conflict Resolution*, 1958.

[57] Kempe, D., Kleinberg, J., and Tardos, É., "Maximizing the Spread of Influence through a Social Network," in *SIGKDD*, ACM Press, 2003.

[58] Khelil, A., Becker, C., Tian, J., and Rothermel, K., "An epidemic model for information diffusion in manets," in *Proceedings of the 5th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*, MSWiM '02, (New York, NY, USA), pp. 54–60, ACM, 2002.

[59] Kim, S., Kim, M., Park, S., Jin, Y., and Choi, W., "Gate reminder: A design case of a smart reminder," in *In the Conference on Designing Interactive Systems*, 2004.

[60] Kimura, M., Saito, K., and Nakano, R., "Extracting influential nodes for information diffusion on a social network," in *Proc. of AAAI*, 2007.

[61] Kimura, M. and Saito, K., "Tractable models for information diffusion in social networks," in *Knowledge Discovery in Databases: PKDD 2006*, Springer Berlin / Heidelberg, 2006.

[62] Kondor, R. I. and Lafferty, J. D., "Diffusion kernels on graphs and other discrete input spaces," in *Proceedings of ICML*, 2002.

[63] Latane, B., "The psychology of social impact," *American Psychologist*, 1981.

[64] Liggett, T. M., *Interacting Particle System.* Springer, 1985.

[65] Liu, L., Bamba, B., Doo, M., Pesti, P., and Weber, M., *mTrigger: An Event-based Framework for Location-based Mobile Triggers.* Information Science Reference, 2009.

[66] Ludford, P., Frankowski, D., Reily, K., Wilms, K., and Terveen, L., "Because I Carry My Cell Phone Anyway: Functional Location-Based Reminder Applications," in *SIGCHI*, 2006.

[67] Ma, H., Yang, H., Lyu, M. R., and King, I., "Mining social networks using heat diffusion processes for marketing candidates selection," in *Proceeding of ACM CIKM*, 2008.

[68] Macy, M. W., "Chains of Cooperation: Threshold Effects in Collective Action," *American Sociological Review*, 1991.

[69] Marmasse, N. and Schmandt, C., "Location-Aware Information Delivery with ComMotion," in *Proc. HUC*, 2000.

[70] Massa, P. and Avesani, P., "Trust Metrics in Recommender Systems," in *Computing with Social Trust*, 2009.

[71] Mohite, M. and Narahari, Y., "Incentive compatible influence maximization in social networks and application to viral marketing," in *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 3*, AAMAS '11, pp. 1081–1082, International Foundation for Autonomous Agents and Multiagent Systems, 2011.

[72] Murugappan, A. and Liu, L., "Energy-Efficient Processing of Spatial Alarms on Mobile Clients," in *Proc. International Conference on Software Engineering and Data Engineering*, 2008.

[73] Nievergelt, J., Hinterberger, H., and Sevcik, K. C., "The grid file: An adaptable, symmetric multikey file structure," in *TODS*, 1984.

[74] Page, L., Brin, S., Motwani, R., and Winograd, T., "The pagerank citation ranking: Bringing order to the web.," tech. rep., Stanford University, 1999.

[75] Peleg, D., "Local majority voting, small coalitions, and controlling monopolies in graphs: A review," in *3rd Colloq. on Structural Information and Communication*, 2002.

[76] Pesti, P., Bamba, B., Doo, M., Liu, L., Palanisamy, B., and Weber, M., "GTMobiSim: A Mobile Trace Generator for Road Networks," tech. rep., Georgia Institute of Technology, 2009.

[77] Pesti, P., Liu, L., Bamba, B., Iyengar, A., and Weber, M., "RoadTrack: Scaling Location Updates for Mobiles on Road Networks with Query Awareness," in *VLDB*, 2010.

[78] Phelps, C. E., "Chapter 5 information diffusion and best practice adoption," in *Handbook of Health Economics* (Culyer, A. J. and Newhouse, J. P., eds.), Elsevier, 2000.

[79] Placek, P. J., "Direct Mail and Information Diffusion," *Public Opinion Quarterly*, 1974.

[80] Prabhakar, S., Xia, Y., Kalashnikov, D. V., Aref, W. G., and Hambrusch, S. E., "Query indexing and velocity constrained indexing: Scalable techniques for continuous queries on moving objects," *IEEE Transactions on Computers*, vol. 51, no. 10, 2002.

[81] Richardson, M. and Domingos, P., "Mining knowledge-sharing sites for viral marketing," in *Proceedings of SIGKDD*, 2002.

[82] Rogers, E. M., "Diffusion of Innovations," *Free Press, New York*, 1995.

[83] Schelling, T., "Micromotives and macrobehavior," *Norton*, 1978.

[84] Seeger, B. and Kriegel, H. P., "Techniques for design and implementation of efficient spatial access methods," in *Proc. VLDB*, 1988.

[85] SIGMOD, *A Generic Framework for Monitoring Continuous Spatial Queries over Moving Objects*, (SIGMOD), 2005.

[86] Skeels, M. M. and Grudin, J., "When social networks cross boundaries: a case study of workplace use of facebook and linkedin," in *Proceedings of ACM GROUP*, 2009.

[87] Sohn, T., Li, K., Lee, G., Smith, I., Scott, J., and Griswold, W., "Place-Its: A Study of Location-Based Reminders on Mobile Phones," in *UbiComp*, 2005.

[88] Travers, J. and Milgram, S., "An Experimental Study of the Small World Problem," *Sociometry*, 1969.

[89] Valente, T., *Network Models of the Diffusion of Innovations*. Hampton Press, 1995.

[90] Watts, D., *Six Degrees: The Science of a Connected Age*. W. W. Norton & Company, 2003.

[91] Weng, J., Lim, E.-P., Jiang, J., and He, Q., "TwitterRank: finding topic-sensitive influential twitterers," in *Proceedings of ACM WSDM*, 2010.

[92] Yang, H., King, I., and Lyu, M. R., "DiffusionRank: a possible penicillin for web spamming," in *Proceedings of ACM SIGIR*, 2007.

[93] Zhou, Y. and Liu, L., "Large graph clustering based on social influence," tech. rep., Georgia Institute of Technology, 2012.

VITA

Myungcheol Doo, who is from Seoul, Korea, received B.S. in Computer Science Education at Korea University, Seoul, Korea in 2004. While he was undergraduate student, he has worked as a software engineer at Hanmidata for 3 years. After graduation from Korea University, he worked at TSIS shortly and moved to Georgia Institute of Technology to purse M.S. and Ph.D. in Computer Science. Myungcheol has pursued his dissertation research in the area of Location-based Services, Spatial Indexing, and Social Influence under the guidance of Prof. Ling Liu. During his graduate program, he has worked at IBM T.J. Watson Research Center and Applied Research Center at Motorola Mobility. His intern works have been published as papers in conferences and submitted as patents.

Besides the academic background, Myungcheol Doo has many leadership experience in various positions. He was the President of Korean Student Association in College of Computing, Georgia Tech from 2006 to 2007 and the secretary of Korean Student Association in Georgia Tech. During the undergraduate at Korea University, he co-founded the alumni of the Young Tigers, which is a freshmen cheer leading squad, at Korea University.